

D3.3 – Concepts and algorithms incl. V&V results from 1st cycle

Project Number:	690705
Classification	Public
Deliverable No.:	D3.3
Work Package(s):	WP 3
Document Version:	Vs. 1.0
Issue Date:	30.06.2017
Document Timescale:	Project Start Date: September 1, 2016
Start of the Document:	Month 08
Final version due:	Month 10
Deliverable Overview:	<p>Main document: D3.3 Concept & algorithms; PUBLIC</p> <p>Annex: no annexes</p>
Compiled by:	M. Graf, ULM
Authors:	M. Graf (ULM), M. Eilers (HMT), S. Suck (OFF), Paulin Pekezou Fouopi (DLR), David Käthner (DLR)
Technical Approval:	Fabio Tango, CRF
Issue Authorisation:	Andreas Lüdtke, OFF

© All rights reserved by AutoMate consortium

This document is supplied by the specific AutoMate work package quoted above on the express condition that it is treated as confidential to those specifically mentioned on the distribution list. No use may be made thereof other than expressly authorised by the AutoMate Project Board.



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

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

Table of Contents

1	Introduction	4
2	Trajectory planning	4
2.1	Environment model and decider module	5
2.2	Sampling based concepts.....	6
2.3	Continuous concepts.....	11
2.4	First results	15
3	Online risk assessment.....	17
3.1	Idea and Metric	17
3.2	Required Input	18
3.3	Provided Output	19
3.4	Calculation of a Safety Corridor	20
3.4.1	Deriving the Safety Regions for the Lane Boundary	21
3.4.2	Deriving the Safety Regions for Objects near the TeamMate Vehicle	23
3.5	Implementation, Verification and Validation for the First Cycle	25
3.6	Potential Improvements for Future Cycles	27
4	Algorithms to learn from the driver	28
4.1	Idea and Required Input.....	28
4.2	Learning Procedure	30
4.3	Verification and Validation for the First Cycle	32
5	Conclusion	33
6	Literature	34

1 Introduction

The overall goal of the AutoMate project is to develop the “Team Mate” car. In addition to many assistance functions, this system will be able to drive autonomously, if a sufficient number of sensor data is available. In order to be able to maneuver safely through the environment, a driving strategy must be available, which at the same time ensures safety and comfort for the vehicle driver, as well as for the other road users. This task is very challenging, as decisions and subsequent actions must be planned for the future. While humans do this task intuitively, the automation has to be based on a quantitative environment model. In addition, the actions of further traffic participants have to be estimated. For this purpose, behaviour models are used for other traffic participants, which provide uncertainty-based estimates regarding positions, speed, as well as other required information. Based on this, a concrete path and corresponding velocities, accelerations etc. can be planned. This process is called trajectory planning. The values contained in the trajectory thus represent the action to be performed for the vehicle and are forwarded to the vehicle controller. In addition, the vehicle shall learn the driver’s preferences. Therefor learning algorithms will be used, which adapt the parameters of the trajectory planning algorithm accordingly.

The above tasks describe the contents of WP3. In this deliverable, possible concepts for trajectory planning and related components will be presented. In addition, first results of the V&V from the first cycle are shown.

The deliverable is structured as follows:

In section 2, common concepts for trajectory planning are presented. The methods are divided into sampling-based and continuous methods. In addition, it is described, how collisions with obstacles of any kind will be avoided. Section 3 contains a description of the risk assessment and the resulting safety corridors. In section 4, offline and online learning algorithms are described.

2 Trajectory planning

In the following, common concepts for trajectory planning are described. Therefor a division between sampling-based and continuous methods is made. First however, necessary requirements are discussed.

2.1 Environment model and decider module

In order to plan trajectories so, that the resulting vehicle behaviour is safe and comfortable, the trajectories must be evaluated with sufficient accuracy. For this purpose, it must be possible to ascertain whether the trajectory is collision-free and does not lead to collisions with other vehicles, pedestrians etc. In addition, it must be possible to check whether the trajectory will keep the vehicle in between the road boundaries. For this purpose, an environment model is required. There are different approaches to model the environment.

The team "MIT" used a "Drivability Map" (Kuwata, Y. et al. 2009). during the DARPA Urban Challenge. This "Drivability Map" is a discrete representation of two-dimensional space. Each cell within the grid carries information, whether the cell is a forbidden zone or not, also cells which are drivable contain costs, to evaluate a trajectory passing these cells.

Figure 1 illustrates an example of the drivability map.

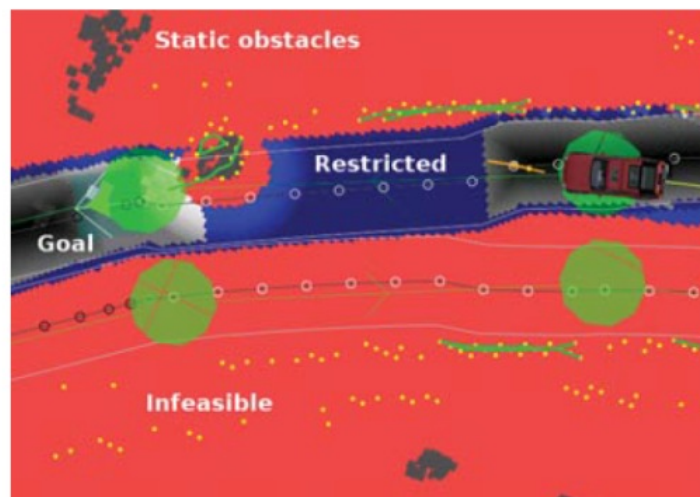


Figure 1: Drivability Map (Kuwata, Y. et al. 2009).

Another possibility for environment modelling, offer so-called "corridors". In (Ziegler J. et al. 2014) such a corridor consists of polygonal lines, representing the boundaries of the lane.

Another requirement is a decision making module, which is preceded to the trajectory planner. Such a module is needed, since traffic situations become arbitrarily complex, so decisions about future actions must be made. This module is intended to describe the "rough" driving method, e.g. a maneuver class. In AutoMate, driving maneuvers are to be specified (refer to deliverable 3.2). The decider module has to find the most appropriate maneuver, based on criteria such as risk assessment etc.

2.2 Sampling based concepts

Rapidly Exploring Random Trees

Rapidly Exploring Random Trees (LaValle, S. M. 1998) (RRTs) were developed at Iowa State University in 1998 and are one of the most widely used methods for trajectory and path planning. RRTs are used not only in autonomous driving but also in other areas of robotics. The basic idea is to scan the state or work space on a random based sampling scheme. The algorithm works as follows:

Suppose you want to plan a trajectory for a vehicle from x_0 to x_F and the vehicle model can be described roughly by the differential equation $\dot{x} = f(x, u)$. First, a random point x_{rand} is generated and the point in the tree is selected, which is closest to x_{rand} with respect to a previously defined metric ρ . Subsequently, control variables u are randomly sampled for the next Δt seconds. For each sample of u , the vehicle model will be integrated. The resulting trajectories must, of course, be checked for collision-freeness. From the remaining trajectories, the one closest to the point x_{rand} with respect to ρ is selected.

The algorithm (refer to LaValle, S. M. 1998) is summarized below:

```
GENERATE_RRT( $x_{init}, K, \Delta t$ )
1   $\mathcal{T}.init(x_{init});$ 
2  for  $k = 1$  to  $K$  do
3       $x_{rand} \leftarrow \text{RANDOM\_STATE}();$ 
4       $x_{near} \leftarrow \text{NEAREST\_NEIGHBOR}(x_{rand}, \mathcal{T});$ 
5       $u \leftarrow \text{SELECT\_INPUT}(x_{rand}, x_{near});$ 
6       $x_{new} \leftarrow \text{NEW\_STATE}(x_{near}, u, \Delta t);$ 
7       $\mathcal{T}.add\_vertex(x_{new});$ 
8       $\mathcal{T}.add\_edge(x_{near}, x_{new}, u);$ 
9  Return  $\mathcal{T}$ 
```

Algorithm 1: RRT- algorithm (LaValle, S. M. 1998).

Figure 2 shows an example of the resulting tree of trajectories or paths.

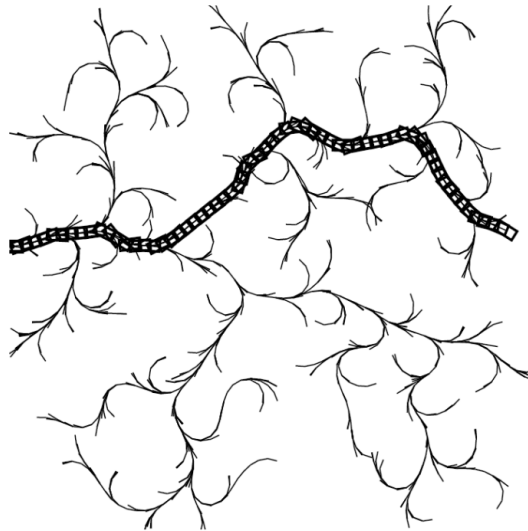


Figure 2: Resulting tree of paths from the RRT-algorithm (refer to LaValle, S. M. 1998).

A tried and tested approach from the class of the RRT algorithms are so-called closed-loop RRTs (Kuwata, Y. et al. 2008) and were used, for example by the MIT team at the 2007 DARPA Urban Challenge.

Instead of the control variables, the setpoint variables for an underlying steering controller are sampled in this approach.

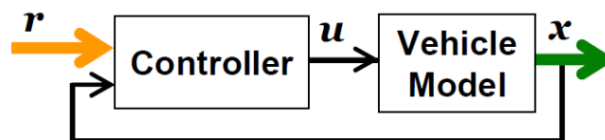


Figure 3: Steering Controller with Vehicle model for closed loop RRT.

The reference path r is determined randomly. With the aid of a steering controller the path r is followed. The vehicle model ensures, that the trajectory will comply with the vehicle dynamics. The sampling of reference paths r yields a tree of trajectories, from which the best is selected. This is illustrated below:

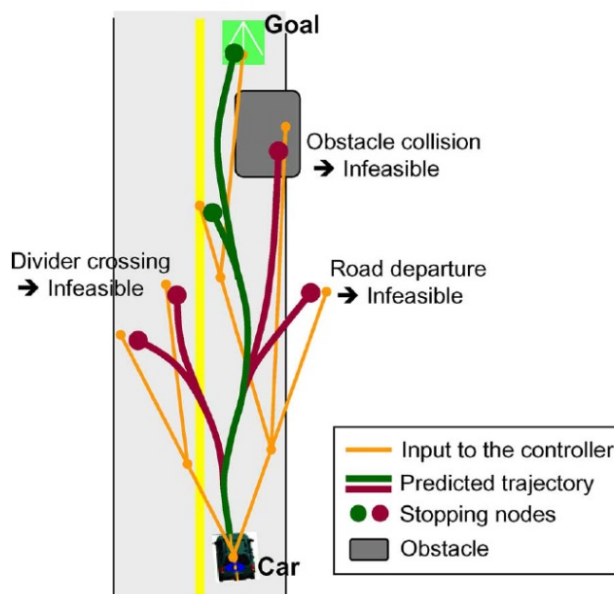


Figure 4: Visualisation of the closed-loop-RRT algorithm.

Finally, the best alternative in the tree of possible trajectories must be found. One possibility for this is to apply tree search algorithms. As an alternative, it is possible to use the RRT* algorithm (Karaman, S et al. 2011) which determines the optimal solution according to a defined cost function.

State Lattices

Another possibility for path planning is the use of so-called "state lattices" (Pivtoraiko, M., Kelly, A., 2005). A tree of previously defined movement primitives is searched. Each combination of these primitives is checked for validity with respect to collision-free and dynamic constraints. Finally, from the remaining combinations, the one which is optimal with regard to a certain cost function is selected, refer to figure 5.

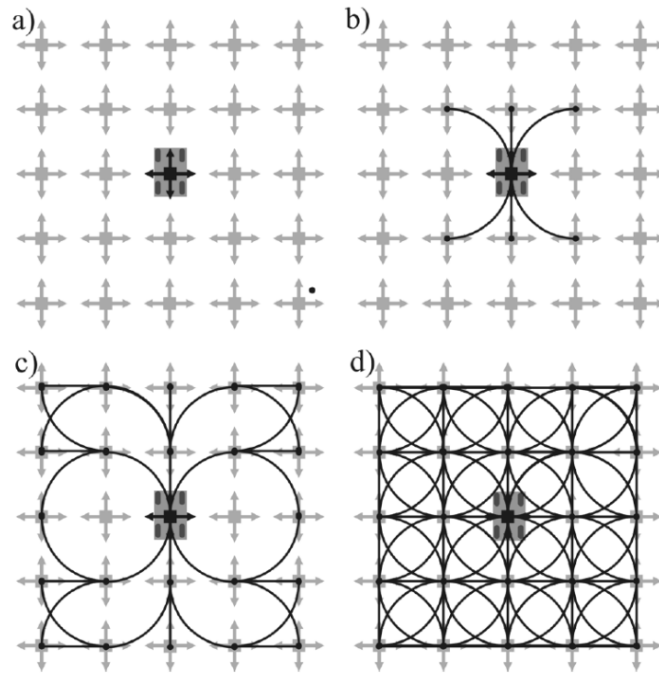


Figure 5: State Lattices approach for path searching (Pivtoraiko, M., Kelly, A., 2005).

The pictures a) –d) show how different positions in the workspace can be achieved by the application of the motion primitives. In addition, it can be seen, that the motion primitives are selected in such a way, that discrete spatial positions (in figure 6 equidistant 2D-points) are achieved.

A canonical set for planning of real vehicle trajectories could for example look like as in figure 6 (Ziegler J. and Stiller C. 2009).

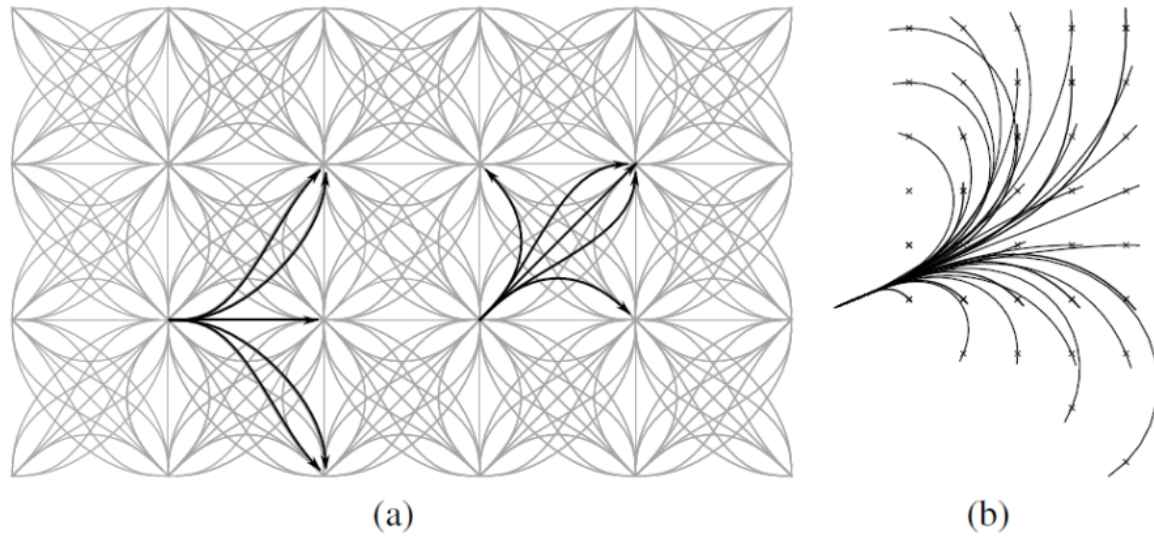


Figure 6: a) Grid resulting from motion primitives. b) Motion primitives for real vehicle trajectories. Refer to (Ziegler J. and Stiller C. 2009).

In figure 6 a) the resulting grid of achievable points can be recognized. In b), the motion primitives can be seen.

The grid shown in figure 6 does not contain any temporal information, thus it can only deal with pure path planning problems. To incorporate temporal information, a discrete set of velocities is selected as shown in Illustration 6.

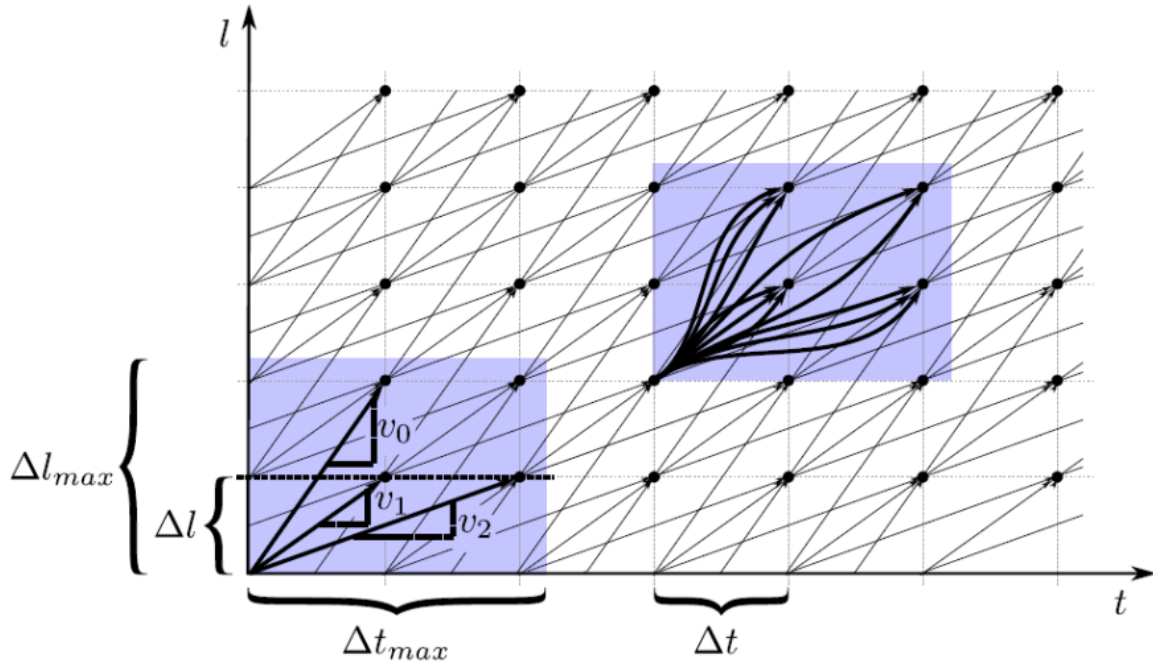


Figure 7: Discrete velocities for the incorporation of temporal information [6].

For the explanation, it is first assumed that only constant velocities are sampled, as shown in Figure 7 below. In this case, the speeds are multiples of the fraction $\Delta l/\Delta t$, in the example $v_0 = 2\Delta l/\Delta t$, $v_1 = 1\Delta l/\Delta t$, $v_2 = \Delta l/\Delta t$. In order to generate a jerk-minimal behaviour, polynomials can be used instead of constant velocities (figure 6 right above).

2.3 Continuous concepts

Potential Fields

The method of the potential fields (see for example, Espitia H. E. and Sofrony J. I. 2011) represents a possibility for continuous path planning. If a path from A to B shall be planned, an attractive potential is assigned to the target point. In order to avoid collisions with obstacles, obstacles are assigned a repulsive potential. The assignment of potential takes place via so-called "potential functions". A potential function for attraction may, for example, be as follows

$$U_{att}(p, v) = \alpha_p \|p_{tar}(t) - p(t)\|^m + \alpha_v \|v_{tar}(t) - v(t)\|^n.$$



Here, a target speed is additionally included, to which the robot is to be accelerated. The path is now obtained by the gradient formation of the potential function, the resulting so-called "virtual force field" can be calculated as follows

$$\mathbf{F}_{att,p} = -\nabla_p U_{att}(\mathbf{p}, \mathbf{v}) = -\frac{\partial U_{att}(\mathbf{p}, \mathbf{v})}{\partial \mathbf{p}}$$

$$\mathbf{F}_{att,v} = -\nabla_v U_{att}(\mathbf{p}, \mathbf{v}) = -\frac{\partial U_{att}(\mathbf{p}, \mathbf{v})}{\partial \mathbf{v}} .$$

Correspondingly, the repulsive force fields results

$$\mathbf{F}_{rep,p} = -\nabla_p U_{rep}(\mathbf{p}, \mathbf{v})$$

$$\mathbf{F}_{rep,v} = -\nabla_v U_{rep}(\mathbf{p}, \mathbf{v}).$$

The absolute force fields, can be obtained by addition

$$\mathbf{F}_{total} = \mathbf{F}_{att} + \mathbf{F}_{rep}.$$

This procedure merely describes the basic principles of the potential field method. There are many modifications which lead to corresponding improvements in various cases.

Optimal Control Planner

An example of a continuous concept for trajectory planning is the method described in (Ziegler J. et al. 2014) Here the trajectory is approximated by 2D points $x_k = x(t_k)$, $k = 1 \dots n$ and linearly interpolated between this points. Velocities, accelerations etc. are determined by differential quotients. The variables x_k now serve as variables of an optimisation problem, which is set up in such a way, that a defined cost function is minimised and certain constraints are statisfied. The Cost function is as follows

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

Where j_{offs} is a cost term, which contributes to drive the vehicle in the middle between the lane's boundary lines.

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

The functions d_{left} , d_{right} are signed distance functions. 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

$$j_{vel}(x(t)) = w_{vel} |v_{des}(x(t)) - \dot{x}(t)|^2,$$

will help to reach the current top speed. The vector v_{des} corresponds to the actual top speed. For more details refer to (Ziegler J. et al. 2014). The acceleration term

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

contributes to minimise the acceleration and the jerk term

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

contributes to minimise the jerk. And finally the term for the yawrate

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

The variables w_i are weighting factors.

To avoid collisions with other vehicles, their positions are predicted for time intervals. Subsequently, obstacle polygons are defined to approximate their shape for time intervals (see figure 8).

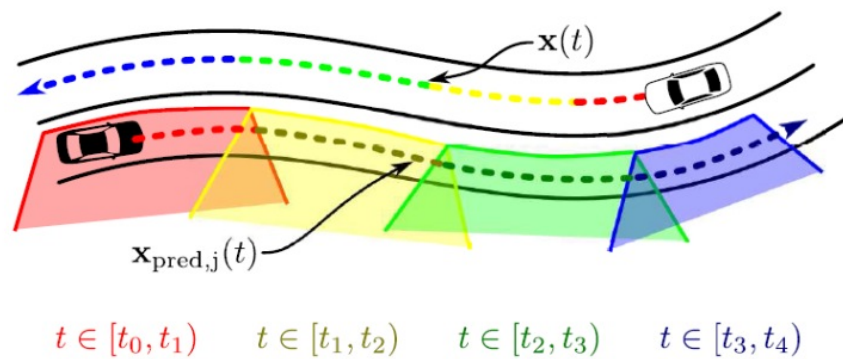


Figure 8: Polygons for collision avoidance with other vehicles (Ziegler J. et al. 2014).

In the next step, the shape of the ego vehicle has to be approximated. For this, circles are used, see figure 9.

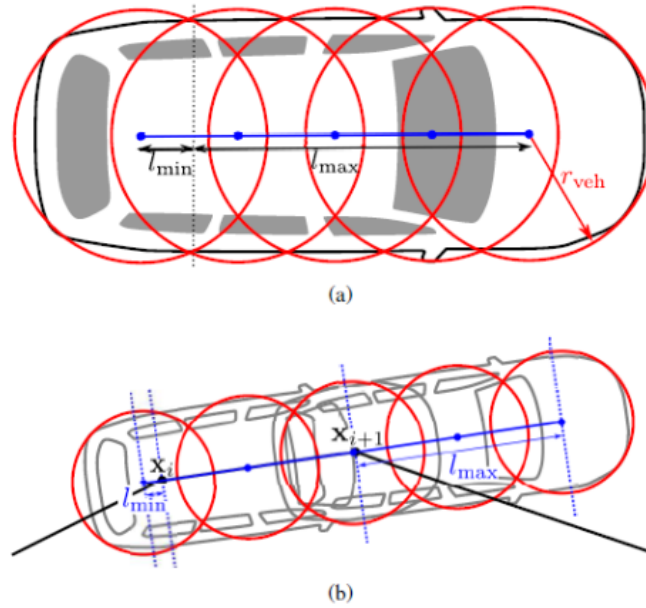


Figure 9: Circles to approximate the vehicles shape (Ziegler J. et al. 2014).

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 center 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.

2.4 First results

As mentioned, this deliverable will present first results. The status of task 3.4, which is mainly about trajectory planning, can be summarized as follows: The concept of the optimal control planner has been selected and programming has also started. For this purpose, a simulation environment was created in Matlab to develop and test the planner on real data. In order to obtain satisfactory results, further errors have to be removed from the code, as well as suitable parameters for costs, etc. have to be found. Figures 10-12 show examples of the preliminary results.

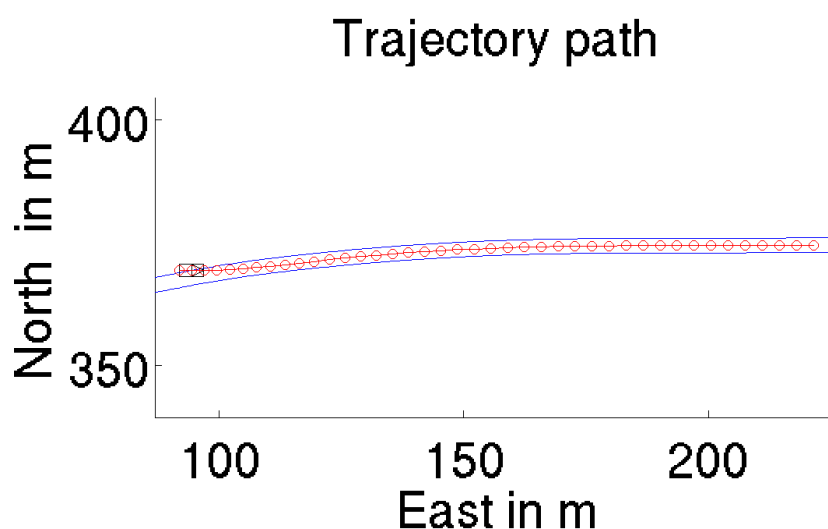


Figure 10: Path of the trajectory.

Trajectory velocity

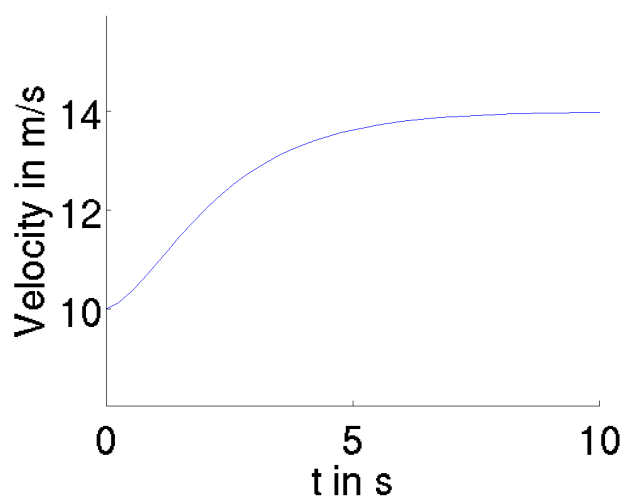


Figure 11: Velocity of the trajectory.

Trajectory acceleration

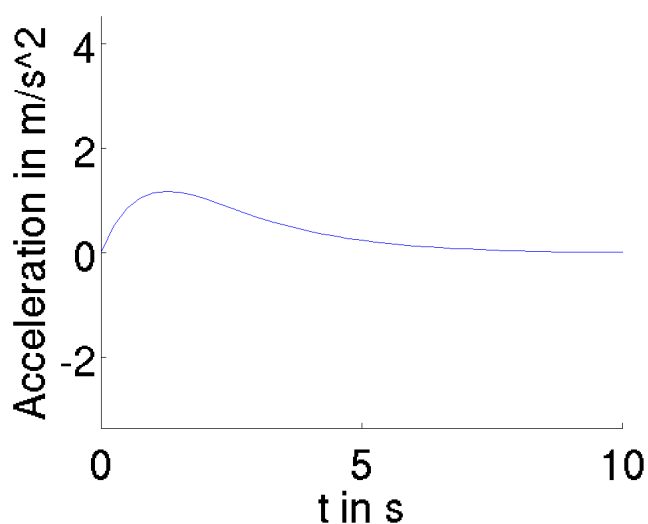


Figure 12: Acceleration of the trajectory

The path of the trajectory can be seen in the figure 10. The vehicle starts with lateral offset to the road and then enters the roadway. The trajectory speed approaches the speed limit of 14m/s as desired. The acceleration required for this purpose remains below 2m/s² and thus does not reach too high values.

3 Online risk assessment

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 will be 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. In Deliverable D3.1 “Metrics and plan for V&V of the concepts and algorithms in the 1st cycle”, we provided a first generic and high level view for online risk assessment. In this deliverable, we will refine this high-level view and provide more concrete definitions of the required input and the provided output, as well as a first description of how the output are obtained in the first cycle of AutoMate.

3.1 Idea and Metric

For the first phase in AutoMate, we implement online risk assessment using an abstract metric for risk based on the *probability of collision*. To provide some intuition, the general idea for online risk assessment in the first cycle is as follows (see Figure 1 for an example):

We generate a set of *safety regions*, one related to the road boundaries, and each one for each traffic participant and obstacle in the vicinity of the TeamMate vehicle. Each safety region defines a region in which the probability of collision between the TeamMate car and the single object corresponding to the safety region is limited by a user-defined threshold δ : $P(\text{Collision between TeamMate vehicle and object}) \leq \delta$. The safety corridor is then defined as the conjunction of these regions. We note that this metric has shortcomings, in that it is possible that $P(\text{Collision between TeamMate vehicle and any object}) > \delta$. We will address this issue in future cycles of AutoMate (c.f. 3.5).

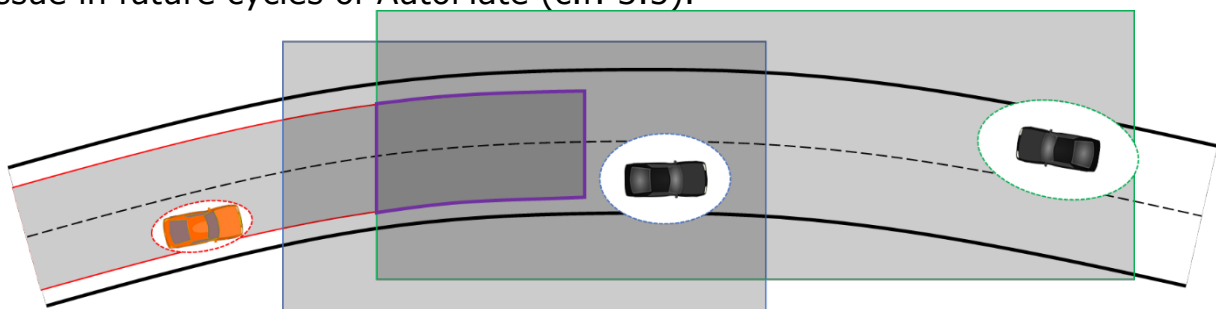


Figure 1: Exemplary visualization of a safety corridor based on a metric for the probability of collision. The safety corridor is the conjunction (purple) of (for visualization purposes arbitrarily cropped) independent safety regions, one for the road boundaries (red), and each one of the other traffic participants, depending on the beliefs about the world. Each safety

region provides an upper bound δ for the collision probability. The safety corridor as a conjunction also provides an upper bound on the collision probability, which however surpasses δ .

3.2 Required Input

For the first cycle in AutoMate, we base online risk assessment on the assumption and requirement that at each point in time t , input is provided in the form of a beliefs about the world in terms of a world model \mathbf{w}^t , and beliefs about the future spatial and temporal evolution of the traffic scene $\mathbf{e}^{t+1:t+n}$ up to a number of n timesteps, in the following simply referred to as *episode* (an visual example is provided in Figure 2).

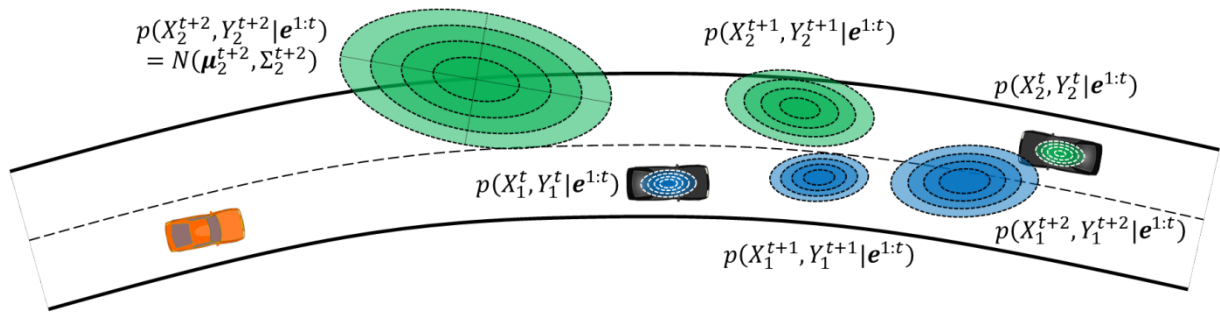


Figure 2: Exemplary visualization of the required input in terms of a world model \mathbf{w}^t and an episode $\mathbf{e}^{t+1:t+n}$. For online risk assessment, we assume that the marginalized belief states over the positions of other traffic participants and objects at different time steps can be reasonable represented as bivariate Gaussian distributions. The TeamMate vehicle is represented by the orange vehicle.

More specifically, we assume that the world model \mathbf{w}^t provides the following information:

- Let $\mathbf{o}^{1:t}$ denote the history of sensor observations and other evidence collected by the TeamMate vehicle up to t . The world model \mathbf{w}^t is assumed to provide a belief state $p(X_{MT}^t, Y_{MT}^t, \theta_{MT}^t | \mathbf{o}^{1:t})$ for the current location of the TeamMate vehicle in terms of Cartesian x- and y-coordinates and its current orientation in terms of a yaw angle θ , in respect to the x-axis.
- Access to a map M , describing the static world. For online risk assessment, this map is assumed to be correct. It is furthermore assumed that at each point in time t , the origin of the map is centred on the mean position of the TeamMate vehicle and the x-axis is aligned with its mean orientation. Due to the uncertainties concerning the TeamMate vehicles true position and orientation, we can accept the mean position and orientation of the TeamMate vehicle as correct, and instead treat corresponding uncertainties as uncertainties concerning the position and orientation of the map.



- Let $\mathbf{v} = \{v_i, \dots, v_m\}$ denote a set of traffic participants and static objects in the vicinity of the TeamMate vehicle. For each object $v \in \mathbf{v}$, the world model w^t is assumed to provide a belief state $p(X_v^t, Y_v^t, \Theta_v^t | \mathbf{o}^{1:t})$ for its position and orientation. Based on the assumption that these beliefs are obtained from the TeamMate's sensor platform, we furthermore assume that the uncertainties about the state of the TeamMate vehicle itself do not affect these beliefs. The world model is furthermore assumed to provide the probability of existence $P(E_v^t | \mathbf{o}^{1:t})$ and belief states about the size for each object $v \in V$ in terms of a conditional probability density function over the width $p(W_v^t | \mathbf{o}^{1:t})$ and length $p(L_v^t | \mathbf{o}^{1:t})$.

Concerning the future spatial and temporal evolution of the traffic scene, for online risk assessment in the first cycle², we assume that an episode $e^{t+1:t+n}$ provides the following information:

- For each object $v \in \mathbf{v}$, an episode $e^{t+1:t+n}$ is assumed to provide belief states for the Cartesian coordinates (either globally or relative to the TeamMate vehicle) of the geometrical centre and the orientation of the object v at certain points in the future $p(X_v^{t+i}, Y_v^{t+i}, \Theta_v^{t+i} | \mathbf{o}^{1:t}), i = 0, \dots, n$ up to a user-defined prediction horizon $n \geq 1$.

3.3 Provided Output

As before, let n denote a desired prediction horizon and m denote the number of objects in the vicinity of the TeamMate vehicle. For the first version of online risk assessment and in respect to the metric introduced in Section 3.1, the output of the online risk assessment is defined as a set $\mathbf{c}^{t:t+n}$ of *safety corridors* $\mathbf{c}^{t:t+n} = (\mathbf{c}^{t:t+1}, \mathbf{c}^{t+1:t+2}, \dots, \mathbf{c}^{t+n-1:t+n})$. Each safety corridor $\mathbf{c}^{i:i+1}, t \leq i \leq n-1$ defines a region over a temporal interval $[i, i+1]$ for which the probability of collision between the TeamMate vehicle and a *single* object $v \in \mathbf{v}$ or the road boundaries is below a set of user-defined thresholds. Formally, each safety corridor $\mathbf{c}^{i:i+1}$ is defined as a set of *polygonal lines* $\mathbf{c}^{i:i+1} = \{L_R^{t:t+n}, L_1^{i:i+1}, \dots, L_m^{i:i+1}\}$, 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 $A_j = (x_j, y_j)$ denoting the x- and y-coordinates in a Cartesian coordinate system.

For a safety corridor $\mathbf{c}^{i:i+1} = \{L_R^{t:t+n}, L_1^{i:i+1}, \dots, L_m^{i:i+1}\}$, $L_R^{t:t+n}$ denotes a polygonal line derived from the road boundaries, that *encloses* a region in which the

² We expect the required input to be extended for future versions of online risk assessment.

probability of collision with the road boundaries is below a threshold δ_R . Each $L_j^{i:i+1}, j = 1, \dots, m$ denotes a polygonal line that *excludes* a region for which the probability of collision with a corresponding object is below a threshold δ_V . A visual example of a safety corridor is provided in Figure 3. We note that a safety corridor abstracts from the dimension of the TeamMate vehicle, which should instead be taken into account by the path planning algorithms.

If required for path planning, a safety corridor $c^{i:i+1} = \{L_R, L_1^{i:i+1}, \dots, L_n^{i:i+1}\}$ can be further reduced to a single polygonal line, enclosing the area of collision-free space, or a set of two polygonal lines representing the left and right boundary of a corridor enclosing the area of collision-free space.

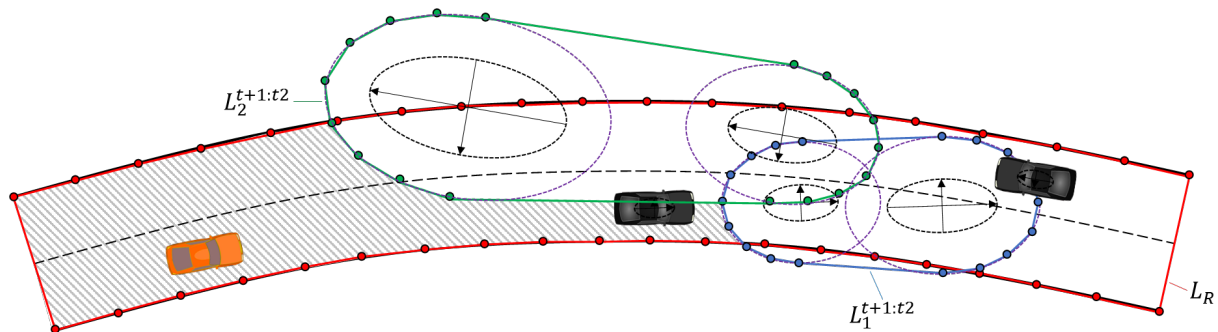


Figure 3: Exemplary visualization of a safety corridor for a temporal interval $[t+1, t+2]$, composed of a polyline $L_R^{t:t+n}$ associated with the lane boundaries and two polylines $L_1^{t+1:t+2}$ (blue) and $L_2^{t+1:t+2}$ (green) associated with two traffic participants. The grey hatched area represents the area of collision-free travel.

3.4 Calculation of a Safety Corridor

Within the first cycle, online risk assessment is implemented as a software component based on the pseudo-code provided in Table 1, i.e., as a function that takes as input the current world model w^t and an episode $e^{t:t+n}$ and provides as output a collection of safety corridors $c^{t:t+n} = \{c^{t:t+1}, \dots, c^{t+n-1:t+n}\}$. The general algorithm can be separated into the calculation of the polygon $L_R^{t:t+n}$ encompassing the region for which the probability of a collision with the road boundary is below the threshold δ_R and the polygons $L_1^{i:i+1}, \dots, L_m^{i:i+1}$, each excluding the region for which the probability of a collision with the respected object is below the threshold δ_V .

Table 1: Pseudo code for online risk assessment in the first cycle of AutoMate.

Algorithm: Online risk assessment

Procedure *derive_safety_corridors* (

```

 $w^t$  // World model for the current time step, including a set of belief
states (omitting the dependence on  $\mathbf{o}^{1:t}$ )
 $p(X_j^t, Y_j^t, \Theta_j^t), p(E_j^t), p(L_j^t), p(W_j^t), i = 0, \dots, n, j = 1, \dots, m$ , where  $m$  denotes
the number of vehicles in the vicinity of the driver, a map  $M$  centred
on the position of the TeamMate vehicle, and a belief state
 $p(X_{TM}^t, Y_{TM}^t, \Theta_{TM}^t)$  concerning position and orientation of the TeamMate
vehicle.
 $\mathbf{e}^{t+1:t+n}$  // Episode for the current time step, including a set of belief states
(omitting the dependence on  $\mathbf{o}^{1:t}$ )  $p(X_j^{t+i}, Y_j^{t+i}, \Theta_j^{t+i}), i = 1, \dots, n, j =$ 
 $1, \dots, m$ , where  $n$  denotes the prediction horizon and  $m$  denotes the
number of vehicles in the vicinity of the driver.
)
1  $L_R^{t:t+n} \leftarrow \text{derive\_lane\_boundary\_polyline}(p(X_{TM}^t, Y_{TM}^t, \Theta_{TM}^t), M)$ 
2 for  $i = 0, \dots, n - 1$ 
3   for each  $j = 1, \dots, m$ , where  $p(E_j^t) \geq \delta_E$ 
4      $L_j^{t+i} \leftarrow \text{derive\_collision\_boundary}(p(X_j^{t+i}, Y_j^{t+i}, \Theta_j^{t+i}), p(L_j^t), p(W_j^t))$ 
5      $L_j^{t+i+1} \leftarrow \text{derive\_collision\_boundary}(p(X_j^{t+i+1}, Y_j^{t+i+1}, \Theta_j^{t+i+1}), p(L_j^t), p(W_j^t))$ 
6      $L_j^{t+i:t+i+1} \leftarrow \text{derive\_convex\_hull}(L_j^{t+i}, L_j^{t+i+1})$ 
7      $\mathbf{c}^{t+i,t+i+1} \leftarrow \{L_R^{t:t+n}, L_1^{t+i:t+i+1}, \dots, L_m^{t+i:t+i+1}\}$ 
8      $\mathbf{c}^{t:t+n} \leftarrow \{\mathbf{c}^{t:t+1}, \dots, \mathbf{c}^{t+n-i:t+n}\}$ 
9   return  $\mathbf{c}^{t:t+n}$ 

```

The different components of the algorithm will be further detailed in the following sub-sections.

3.4.1 Deriving the Safety Regions for the Lane Boundary

As a first step, a polyline needs to be derived from the map M in respect to the current beliefs about the TeamMate vehicle $p(X_{TM}^t, Y_{TM}^t, \Theta_{TM}^t | \mathbf{o}^{1:t})$. In the case of the ULM demonstrator, it is assumed that such a polyline is readily available and provided as additional component of the world model w^t . For the demonstrators without such a road boundary polyline, we implemented the system in Figure 4 for the first cycle of the project. The system consists of 3 Modules: the *Map Extractor*, the *Map Matching* and the *Road Boundary Polyline Generator*.

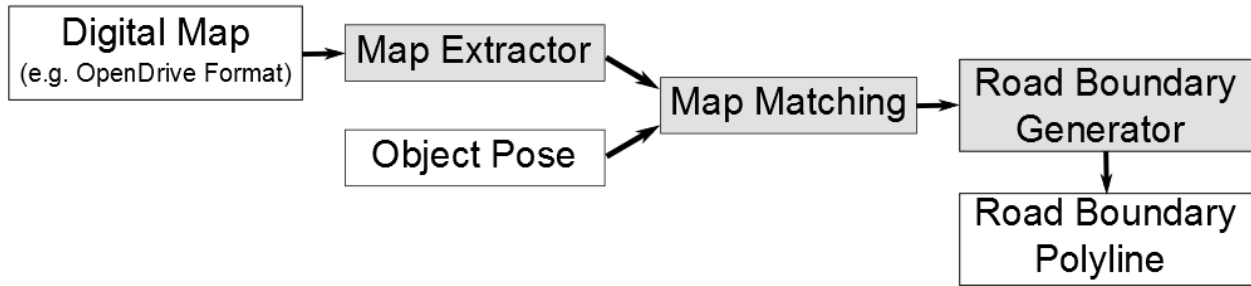


Figure 4: Overview of the system for road boundary ployline generation

The map extractor extracts the map data modelled using the *OpenDrive* format. The extracted map data are saved into a data structure following the *OpenDrive* specification. The *OpenDrive* specification models the road as a set of lanes. Each road has a reference line centred in the middle of the road. The reference line can be modelled as a straight line, a spiral or a cubic polynom, etc. Each lane belonging to the road is parameterized using an offset to the reference line of the road. More information about the lane as the type of the lane marking and the lane type (e.g. driving, emergency, etc.) is provided. Further information on the *OpenDrive* specification can be found in (Dupuis et al. 2015)

After extracting the map data, the ego-vehicle pose is integrated into the map. For this purpose, the lane with the minimal distance to the ego-vehicle is set as the ego-lane. The road associated to the ego-lane is set as the ego-road. Next, the ego-road boundary polyline $L_R^{t:t+n}$ is estimated by sampling from the road border parametric equation in the driving direction with a constant sampling step. Figure 5 shows an example of the generated road boundary polyline projected into the camera image. The road boundary from the map (blue polyline) doesn't accurately match the road border in the camera image because of sensor calibration and ego-pose estimation errors.

In the second cycle of the project, we will address the uncertainties generated by the sensor calibration and ego-pose estimation. Furthermore, we will combine the map data with information generated by the ego vehicle sensors and integrate the generated road boundary polyline into the safety corridor for mobile objects. The safety corridor for mobile objects is described in details in the subsection 3.4.2.



Figure 5: example of road boundary polyline for the ego-vehicle projected into the camera image (see blue polyline)

3.4.2 Deriving the Safety Regions for Objects near the TeamMate Vehicle

For each safety corridor $c^{t+i:t+i+1}$ and each object $v \in \mathcal{V}$, we need to derive a polygonal line $L_v^{t+i:t+i+1}$ that excludes a region for which the probability of collision between the TeamMate vehicle and an object v within the temporal interval $[t+i, t+i+1]$ is below a threshold δ_v . We propose to derive $L_v^{t+i:t+i+1}$ from two belief states $p(X_v^{t+i}, Y_v^{t+i} | \mathbf{o}^{1:t})$ and $p(X_v^{t+i+1}, Y_v^{t+i+1} | \mathbf{o}^{1:t})$ provided by the episode $e^{t+1:t+n}$. For the first version of online risk assessment, we assume that each belief state $p(X_v^j, Y_v^j | \mathbf{o}^{1:t}), t \leq j \leq t+n$ provided by the episode $e^{t+1:t+n}$ will be represented as a bivariate normal distribution, such that $p(X_v^j, Y_v^j | \mathbf{o}^{1:t}) = \int p(X_v^j, Y_v^j, \theta_v^j | \mathbf{o}^{1:t}) d\theta_v^j = \mathcal{N}(\mu_v^j, \Sigma_v^j)$.

Based on the covariance Σ_v^j , we can derive an elliptical contour, that encloses $(1 - \delta_v) * 100\%$ of the probability mass concerning the location of the geometrical centre of an object v . More specifically, let $x = (x, y)^T$ denote a Cartesian coordinate, the interior of an ellipse that encompass $(1 - \delta_v) * 100\%$ of the probability mass of the Gaussian is specified by the coordinates that satisfy $(x - \mu)^T \Sigma^{-1} (x - \mu) \leq \chi_2^2 (1 - \delta_v)$. For deriving the contour, we perform an eigen-decomposition of the covariance matrix $\Sigma_v^j = U_v \Lambda_v U_v^T$, such that the columns of U_v are (normalized) unit eigenvectors and Λ_v is a diagonal matrix of the eigenvalues.

Let's assume that we have an elliptical contour that encloses $(1 - \delta_v) * 100\%$ of the probability mass concerning the location of the geometrical centre of an object v . To take into account the potential size of the object, the contour needs to be augmented by the corresponding size of the object. For now, this will be done by extending the eigenvalues according to the length of the vector given by the length and width of the vehicle, e.g. by using the maximum of the $z_{0.95}$ quantile of both our beliefs about the length $p(L_v^t | \mathbf{o}^{1:t})$ and width $p(W_v^t | \mathbf{o}^{1:t})$.

Lastly, the resulting augmented contour is approximated by a polyline L_v^j by calculating a set of equidistant points on the augmented density contour. Such a polyline L_v^j now approximately encloses an area, such that we would expect a collision outside of the area with δ_v . Performing this operations for both $p(X_v^{t+i}, Y_v^{t+i} | \mathbf{o}^{1:t})$ and $p(X_v^{t+i+1}, Y_v^{t+i+1} | \mathbf{o}^{1:t})$ results in two polylines L_v^{t+i} and L_v^{t+i+1} (an example is provided in Figure 6).

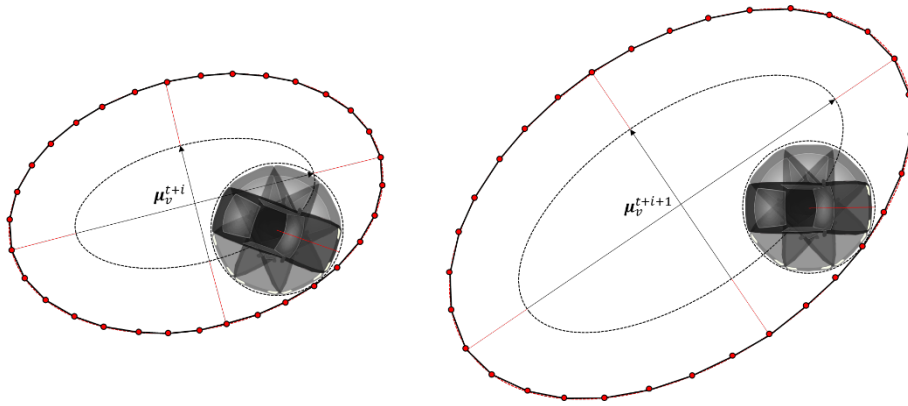


Figure 6: Resulting polylines L_v^{t+i} and L_v^{t+i+1} , approximating the region of collision for a vehicle $v \in V$ at the boundaries of a temporal interval $[t + i, t + i + 1]$.

In the next step, the two polylines L_v^{t+i} and L_v^{t+i+1} need to be combined into a single polyline $L_v^{t+i:t+i+1}$ that approximates the density contour of the temporal interval $[t + i, t + i + 1]$. For the first version of the online risk assessment, this is done by calculating the convex hull of both polylines (Figure 7).

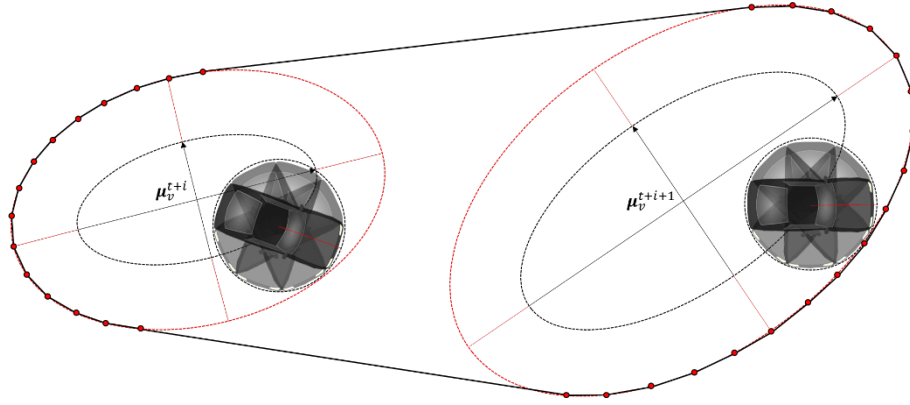


Figure 7: The polyline $L^{t+i:t+i+1}$ approximating the potential locations of a vehicle $v \in V$ within a temporal interval $[t+i, t+i+1]$, derived as the convex hull of L_v^{t+i} and L_v^{t+i+1} .

The safety corridor $c^{i:i+1}$ for the temporal interval $[i, i+1]$ is then composed of the polygon $L_R^{i:i+1}$ enclosing the region for which the probability of a collision with the road boundary is below the threshold δ_R and the polygons $L_1^{i:i+1}, \dots, L_m^{i:i+1}$, each excluding the region for which the probability of a collision with the respected object, such that $c^{i:i+1} = \{L_R^{i:i+1}, L_1^{i:i+1}, \dots, L_m^{i:i+1}\}$. If required for path planning, each safety corridor $c^{i:i+1} = \{L_R, L_1^{i:i+1}, \dots, L_n^{i:i+1}\}$ can be further reduced to a single polygonal line, enclosing the area of collision-free space, or a set of two polygonal lines representing the left and right boundary of a corridor enclosing the area of collision-free space.

3.5 Implementation, Verification and Validation for the First Cycle

As of now, online risk assessment has been implemented for surrounding objects under the assumption that the necessary input can be provided (Figure 8). The corresponding functionality of online risk assessment has been tested using inputs provided by the SILAB simulation environment used at the OFFIS Institute for Information Technology, with episodes obtained using the Constant Yaw and Acceleration (CYRA) motion model (described in D2.2 "Sensor Platform and Models incl. V&V results from 1st cycle").



Figure 8: Screenshot of an exemplary visualization of online risk assessment. The coloured polygons represent the safety corridor for correspondingly coloured vehicles in the vicinity of the TeamMate vehicle (white rectangle) for the temporal

interval $[t + 1s, t + 2s]$. Blue lines indicate heading vectors, the purple line represents the centerline of a two-lane motorway.

For validation purposes, we tested the preliminary correctness of online risk assessment on data sets obtained in driving studies in the SILAB simulation environment. The data set comprises 295123 training samples, recorded with a frequency of 60Hz, with each sample representing the necessary input of up to eight vehicles in the vicinity of the TeamMate vehicle, maximal two vehicles on the current and adjacent lanes, both in front and behind the TeamMate vehicle.

At each time step t , online risk assessment was used to compute the safety polygons for each vehicle in the vicinity of the TeamMate vehicle and ten prediction intervals $[t + 0s, t + 1s], [t + 1s, t + 2s], \dots, [t + 9s, t + 10s]$. At each subsequent time step $t + n$, we then checked, whether the vehicle was inside the corresponding safety polygon estimated by the online risk assessment. For this, we defined that a vehicle was inside a safety interval, if all four corners of the bounding box of the vehicle were located inside the safety polygon. The resulting data was aggregated over all different vehicles to derive the error-rates for each temporal interval. The results are summarized in Table 2. We note that limited (simulated) sensor range of $\pm 200m$ for the detection of surrounding vehicles make it possible that a vehicle was outside the sensor range prior to entering temporal intervals.

Table 2: Validation results of online risk assessment for different temporal intervals using a threshold $\delta_v = 0.05$, aggregated over all vehicles.

Temporal Interval	Vehicles outside of safety interval [#]	Vehicles inside of safety interval [#]	Error-Rate [%]
$[t + 0s, t + 1s]$	56861	75131197	0.075625
$[t + 1s, t + 2s]$	607406	72958919	0.825658
$[t + 2s, t + 3s]$	2624724	69487580	3.63977
$[t + 3s, t + 4s]$	6717195	64099287	9.48536
$[t + 4s, t + 5s]$	11147895	58489173	16.0086
$[t + 5s, t + 6s]$	14870628	53679017	21.6932
$[t + 6s, t + 7s]$	18169861	49339793	26.9145
$[t + 7s, t + 8s]$	21305197	45174097	32.0479
$[t + 8s, t + 9s]$	24423167	41038227	37.3093
$[t + 9s, t + 10s]$	28100175	36565631	43.4545

As apparent from Table 2, the error-rate quickly rises with extended temporal horizon and exceeds the selected threshold for the probability of collision $\delta_v = 0.05$ for temporal intervals above 3 seconds. We expect these results to be improved with the improvement of the underlying vehicle



models. We can trivially improve the performance of the online risk assessment by extending the size of the safety regions. As such, validation must be counterbalanced by a simultaneous maximization of the free space. We will address this issue in the next cycle.

Concerning verification, this document has defined

- the necessary input for online risk assessment in terms of a world model w^t ,
- the definition of the metric and corresponding output of online risk assessment in terms of a set of safety corridors $c^{t:t+n} = \{c^{t,t+1}, \dots, c^{t+n-i,t+n}\}$,
- and a basic overview of the expected implementation of online risk assessments in terms of algorithms.

3.6 Potential Improvements for Future Cycles

Although the implementation of online risk assessment as described in this deliverable is not completed yet, there exist several potential improvements, to be addressed in future cycles of AutoMate.

First and foremost, it should be apparent that the quality of online risk assessment primarily depends on the quality of the provided input, i.e., the world state w^t and especially the episode $e^{t+1:t+n}$. Under the assumption that the episode is derived using vehicle-, situation-, and driver-models developed in WP2, any improvement concerning these models in respect to the prediction of the future evolution of the traffic scene, will consequently improve the performance of online risk assessment.

When assuming the quality of the episode as fixed, online risk assessment could be improved by a better approximation of the safety corridors itself. For the first cycle, the belief states $p(X_v^j, Y_v^j | o^{1:t})$ for an object $v \in V$ are approximated by a bivariate Gaussian distribution and the safety region will be extended without considering potential beliefs about the orientation of the object given its location $p(\Theta_v^j | X_v^j, Y_v^j, o^{1:t})$. For future cycles, we will test to both incorporate the beliefs $p(\Theta_v^j | x_v^j, y_v^j, o^{1:t})$ for a given points on the density contour and to replace the density contour estimation by a density estimation based on sampling from the extended belief $p(X_v^j, Y_v^j, \Theta_v^j | o^{1:t})$. We additionally note that using the convex hull to approximate a polyline $L^{t+i:t+i+1}$ may be a poor approximation of the safety region that allow for potential improvements, e.g. by a linear interpolation of the belief states coupled with an approximation of a concave hull.



Lastly, the currently proposed metric as described in Section 3.1 has flaws. As each safety region only guarantees (in respect to the correctness of assumptions and beliefs) a collision free region for a single reference object, the overall probability of a collision for the conjunction follows a binomial distribution. As such, let X denote the number of objects (including the road boundaries provided by the map as a single object) within the safety corridor, δ denote a global threshold of collision probability and n denote the number of traffic participants and obstacles (resulting in $n + 1$ safety regions due to the road boundaries), the probability that neither the road boundary nor any object is located within the safety corridor is lower-bounded by $P(X = 0) = \binom{n+1}{0} \delta(1 - \delta)^n$, which for a threshold of $\delta = 0.05$ and $n = 10$ traffic participants would be only $P(X = 0) = 0.569$. In future versions, we will try to refine this metric and the notion of safety corridors to provide better lower-bounds. A potential solution could be to interpret the independent beliefs about the traffic participants as a mixture of Gaussians and obtain a single safety region based on density estimation.

4 Algorithms to learn from the driver

The purpose of the algorithms to learn from the driver in AutoMate 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 will learn from the driver during the first cycle of AutoMate.

4.1 Idea and Required Input

In the first cycle learning from the driver is understood as the learning of driving intentions e.g. "*lane change left*". The learning algorithms rely on the probabilistic driver model developed by OFF which creates estimations about the intentions and future behaviour of the human drivers, namely the *Driver Intention Recognition* (DIR) model. This model is described in the corresponding deliverable D2.2.

Since the intentions of the driver can't be observed directly, for the DIR model it is assumed that the intentions of the driver can be interpreted as a hidden process. This process "emits" effects on the traffic situations which are observable. These observable effects can be the 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 in general models the joint density distributions over actions, intentions and observations over an arbitrary length $T \geq 1$ as:



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

Where $\mathbf{O} = \{O_1, \dots, O_{n_O}\}$ denotes a set of continuous and/or discrete random variables that represent the observations of the current traffic situation, $\mathbf{A} = \{A_1, \dots, A_{n_A}\}$ denotes a set of continuous and/or discrete random variables that represent the actions of the driver, and $\mathbf{I} = \{I_1, \dots, I_{n_I}\}$ 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 already be learned from multivariate time-series of driving data via machine-learning methods. To do so the necessary is usually gathered from many different drivers. Since the current offline learning algorithms can't handle datasets with missing values the dataset has to be annotated by an expert. This means for each time point of a sequence of driving situations, described by \mathbf{O}^t and \mathbf{A}^t the human expert labels the situation with the corresponding value of \mathbf{I}^t . So, the driver's intention is not hidden or missing in training data and the model can be learned with *complete data*. After the training process the model in general is able to recognize the driving intentions of human drivers. Since the model was trained with data from several drivers the recognition performance might be suboptimal for the individual driver, e.g. too aggressive for very defensive drivers.

In cycle one, the algorithms to learn from the driver will perform an online learning to recalibrate the parameters of the DIR model. So the model is adjusted to the individual driver during the driving process to match the individual driving behaviour and driving preferences.

Thus the learning algorithms require the same inputs as the DIR model or at least a complete description of the current traffic situation in terms of the world model mentioned in Section 3.2.

For online learning the training data is not annotated by a human expert. So the corresponding driving intentions remain hidden and the algorithm has to handle *incomplete data*.

During driving the online learning will receive for each point in time t the current observation of the traffic situation \mathbf{o}^t (or also world model). Since we are assuming that each manoeuvre happens due to a driving intention the description of the driving situation should contain all information to tell if a certain manoeuvre just happened. For example, to learn the driving intention for a lane change we need to be able to determine when the ego vehicle actually changed its current lane, e.g., $current_lane^{t-1} \neq current_lane^t$.

Additionally an interface to a risk assessment instance is needed in a way that each observed traffic situation could be label with a corresponding risk



value. So, it could be ensured that manoeuvres which contain too risky traffic situations are not considered and learning of unsafe behaviour is avoided.

4.2 Learning Procedure

Assuming a lane change is observed at time t ($current_lane^{t-1} \neq current_lane^t$) the intention I^t is known. Since an actual lane change manoeuvre takes multiple time steps and an intention is usually formed even earlier, we cannot assume that whenever we are not observing that the lane was actually changed that the intention is not "*lane change*". Instead we need to estimate how many previous time steps x , which were observed before t , should also have the intention I^t .

The problem can be formulized as the following smoothing problem:

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

For the online learning of the DIR model the driver intention is a hidden variable and its value is missing for training, except for the moment when the change of lanes is actually observed. Thus, 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. We assume that for the learning every observation there can be only one corresponding intention, so *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 $Y = \{y_1, \dots, y_n\}$, 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. $(y_1, \dots, y_n) := \underset{y_1, \dots, y_n}{\operatorname{argmax}} P_\theta(x_1, \dots, x_n, y_1, \dots, y_n)$
 - b. $\theta := \underset{\theta}{\operatorname{argmax}} P_\theta(x_1, \dots, x_n, y_1, \dots, y_n)$

Point *a* is called the E-step and is the evaluation of the expectation given the current parameter set θ . For using *hard EM* our 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 (Bilmes J. 1998).

An essential part of the online learning the updating of the parameters of the previously offline learned model. Let the observations X and the hidden states Y model a joint density distribution similar to the DIR model. So, an observation at a certain time x^t depends only on the hidden state at the same point in time y^t . While the hidden state y^t only depends on the state at the previous time step y^{t-1} . The nodes of the resulting Bayesian Network at a the current time slice can be described by parameters of the form $\theta_{y|y_i}$ and $\theta_{x|y_i}$. Where $\theta_{y|y_i}$ expresses the probability that y^t takes a certain value if y^{t-1} had the state i . Analogical $\theta_{x|y_i}$ expresses the probability that x^t takes a certain value if y^t is in state i . For the case of Boolean variables each parameter can be described by a Beta distribution. If a variable is non-Boolean a Dirichlet distribution can be used.

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.

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

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. 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 for inference the actual probability can then be computed via the Bayes-Estimate (Koller D. and Friedman N, 2009):

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

4.3 Verification and Validation for the First Cycle

For the validation of the online learning a first implementation was applied to a data set containing about 19 minutes of highway driving with multiple lane changes. The model for recognizing the intention was simplified just to demonstrate the function of the algorithm. Thus it only relies on the observation of the lateral position of the ego car on the current lane, where the deviation from the lane centre is represented by the absolute value $EGO_LAT_POS = [0, 1.85]$. The figures show the distribution for the parameter $\Theta_{EGO_LAT_POS|I=lane_change}$. Figure 7a represents the distribution from the offline learned model while Figure 7b visualizes the distribution after multiple new observations and steps of online learning. The new observations have changed the distribution of the parameter.

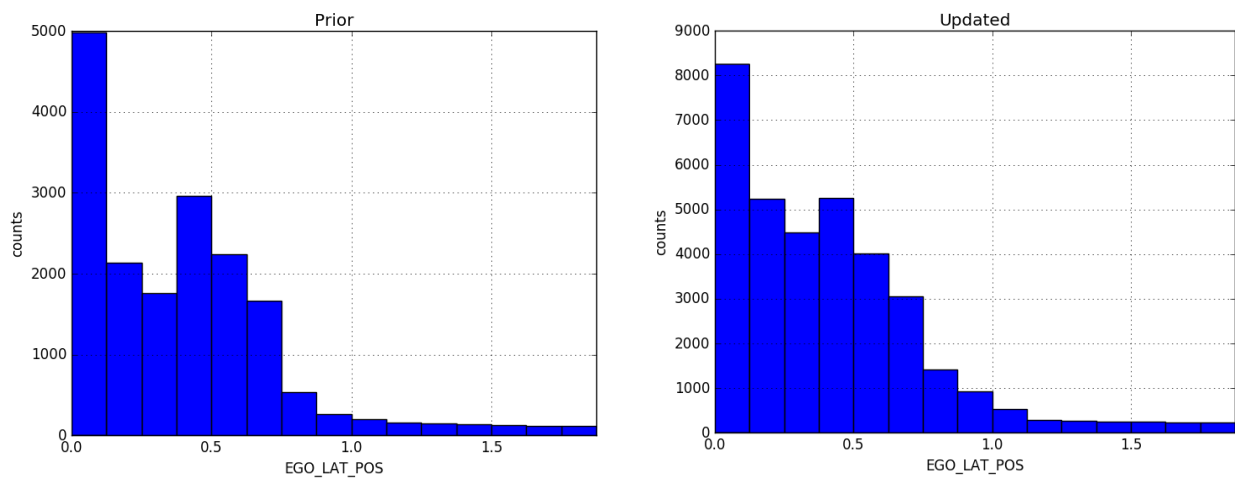


Figure 9: a) Prior distribution of $\Theta_{EGO_LAT_POS|I=lane_change}$, b) updated distribution

Data of manual and automated driving of an AutoMate scenario for learning and validation of the algorithms and models is gathered during June and July 2017 in the driving simulators of ULM and OFF.

5 Conclusion

In this deliverable, concepts for trajectory planning as well as the therefor necessary risk assessment were given. In addition a concept was presented with which it is possible to learn online (while driving) from the driver. In the course of trajectory planning, the vehicle states are to be planned over a specific time horizon. For this purpose, the concept of the optimal control planner described in section 2.3 has been selected. The programming of this concept has already begun. First results are depicted in section 2.4.

Concerning online risk assessment, metrics and algorithms for the estimation of safety corridors in the first cycle of AutoMate have been defined in Section 3 and have been implemented for the SILAB simulation environment. First validation results are presented in Section 3.5. The online learning will use the data, which is obtained during driving to update the Bayesian driver model from WP2. Thus, the general driver model is adapted online to fit the individual driver. Concepts for determining the actually observed behaviour and model parameter updating are described in section 4.2. The implementation is ongoing. A first simplified example for updated model parameters is depicted in section 4.3.

In the second cycle of the project, the implemented trajectory planning algorithm will be first integrated in the ULM demonstrator and tested in some simple traffic scenarios. In order to make the automation as autonomous as possible, a decider module will be switched in front of the trajectory scheduler. This decider module will be used for the parameterization of the planner according to the current scenario. Furthermore, the safety corridors extracted from the road boundary (Section 3.4.1) and dynamic objects (Section 3.4.2) will be merged and furthermore improved based on potential improvements of the metrics and notion of safety corridors (Section 3.6). The interface to the trajectory planning module will be specified and implemented. The online learning will be integrated in the respective demonstrator. It will also be modified to be able to update distributions over continuous variables of the driver model. The interface to the online risk assessment will be defined and integrated to effectively avoid learning from unsafe behaviour.



6 Literature

LaValle, S. M. 1998b. Rapidly-exploring random trees: A new tool for path planning. Report No. TR 98-11, Computer Science Department, Iowa State University. Available at <http://janowiec.cs.iastate.edu/papers/rrt.ps>.

Kuwata Y., Teo J., Karaman S., Fiore G., Frazzoli E., and How J. P., "Motion planning in complex environments using closed-loop prediction," presented at the AIAA Guid., Nav., Control Conf. Exhibit, Honolulu, HI, Aug. 2008, AIAA-2008-7166.

Kuwata Y., Teo J., Fiore G., Karaman S., Frazzoli E. and How J. P., "Real-Time Motion Planning With Applications to Autonomous Urban Driving," in *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105-1118, Sept. 2009.

Karaman, S.; Walter, M.R.; Perez, A.; Frazzoli, E.; Teller, S. (2011): Anytime Motion Planning using RRT*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Pivtoraiko, M., Kelly, A., 2005. Efficient constrained path planning via search in State Lattices. In: International Symposium on Artificial Intelligence, Robotics, and Automation in Space, pp. 1-7.

Ziegler J. and Stiller C., "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," *2009 IEEE/RSJ*

International Conference on Intelligent Robots and Systems, St. Louis, MO, 2009, pp. 1879-1884. doi: 10.1109/IROS.2009.5354448

M. Werling, J. Ziegler, S. Kammel and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame," *2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK, 2010, pp. 987-993.

Ziegler J., Bender P., Dang T. and Stiller C., "Trajectory planning for Bertha — A local, continuous method," *2014 IEEE Intelligent Vehicles Symposium Proceedings*, Dearborn, MI, 2014, pp. 450-457.

Espitia H. E. and Sofrony J. I., "Path planning of mobile robots using potential fields and swarms of Brownian particles," *2011 IEEE Congress of Evolutionary Computation (CEC)*, New Orleans, LA, 2011, pp. 123-129.

Bilmes J., "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models", 1998.

Koller D. and Friedman N., "Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning", 2009, The MIT Press.

Marius Dupuis, "OpenDRIVE. Format Specification", Rev. 1.4, <http://www.opendrive.org/docs/OpenDRIVEFormatSpecRev1.4H.pdf>