Lot4 : Service navettes longue distance
R4.1 : Rapport sur le système complet

Programme	FUI23
Référence	L4.1
Version	1.0
Date	23 / 02 / 2021
Porteur	Renault
Auteur(s)	V. Milanes
Contributeurs(s)	R. Asghar, S. Masi, Ph. Bonnifait, Ph. Xu

Financé par

bpifrance îledeFrance La Région Occitanie L'EUROPE S'ENGAGE en région Auvergne-Rhône-Alpes UNION EUROPÉENNE La Région Auvergne-Rhône-Alpes Yvelines Le Département

Pôles de labellisation

aerospace valley mov'eo Imagine mobility ViaMéca

Contents

List of Figures

1 Introduction

This deliverable presents the technical development carried out for each of the three long distance experimental platforms (based on a Renault ZOE vehicle) to make them operational for the Tornado project. The operation of the technological bricks, emphasising the ones that were especifically developed/modified because of the Rambouillet Operational Design Domain (ODD) conditions are included.

1.1 Previous autonomous driving demonstrations

Automated vehicles progress has significantly advanced in the last years, moving from the first Advanced Driving Assistance Systems (ADAS) as lane departure warning [2] or cruise control systems [3] to highway lateral and longitudinal automated vehicle control as Tesla AutoPilot system. This pioneering production system has demonstrated the ability to introduce advanced supervised technologies in commercial vehicles. L2-based automated highway systems represents the first step toward automated vehicles in more complex traffic scenarios, demanding more intelligent systems.

Urban and rural areas remain as the most challenge scenarios because of the potential changing situations that can be found, differing for well-structured highway scenarios [4]. There are just a few worldwide demonstrations dealing with these complex environments. VisLab's BRAiVE autonomous car carried out a demonstration [5] through Parma streets in Italy mixing rural, highway and urban driving scenarios. Results highlighted that complex scenarios as roundabouts are still an open research challenge. Also in Europe, a cooperation between Daimler and Karlsruhe Institute of Technology (KIT) showed the ability to deal with complex situations in structured environments. Results of the autonomous trip from Mannheim to Pforzheim in Germany were presented in Ziegler et al. [6]. Similar to BRAiVE demonstration, they concluded that overall performance is significantly downgraded with respect to a human driver. Multiple private companies in the United States as Waymo, Zoox or Aptiv are testing their systems in limited well-structured urban environments but there are no scientific papers providing details in the approach.

Rural areas have received little attention [7] but they represent an excellent test bench for automated driving technologies. Conditions are usually degraded with respect to urban zones (i.e. poor or non-line marking available, patched pavement or narrowed roads among others) and they account for more than 97% of the United States land area [8], highlighting the importance of demonstrating (and validating) automated technologies on these zones.

This deliverable presents the results of an automated vehicle demonstration in Rambouillet (France) within the framework of the Tornado project, covering

urban and rural areas.

1.2 Document organization

The deliverable is divided in three main blocks accordingly to the three different platforms developed by project partners. Second section covers Renault development, third section includes Inria contributions and the system studied at Heudiasyc UTC-CNRS is described in section fourth.

2 Renault system

This section describes the Renault autonomous driving system capabilities. It includes the in-vehicle architecture for providing automated vehicle capabilities, focusing later on the development of perception, planning and control stages to overcome the technical challenges. Firstly, details the key driving scenarios encountered during the two-week demonstration

2.1 Introduction

Renault autonomous driving system benefits from a modular architecture whose origins come from a VALET service project demonstrated during the 22nd ITS World Congress held in Bordeaux. A Renault Fluence ZE was fully robotized and equipped with sensors for performing automated parking. A remote emergency braking system was included in the vehicle for safety reasons. The basis of that architecture has been also used here in the Tornado project, keeping its modularity. Next section describes the major challenges that we faced for adapting the system to the Rambouillet ODD.

2.2 Technical challenges

Peri-urban and rural areas present multiple challenging situations. Accordingly to the complexity, the associated challenges are divided as follows:

- **Nominal driving** corresponds to regular traffic situations. It includes standard width lanes, good lane-marking quality or well-identified pedestrian crossings. The way of handling these situations matches with the general architecture description and it serves as starting point for all specific use case scenarios.
- **Roundabout driving** is getting more and more popular replacing crossroads because it increases traffic fluidity, reducing 37% overall collisions [9]

Figure 1: Tornado project key scenarios demonstration. Left graphic corresponds to a roundabout entrance. Centre graphic depicts a narrow two-way road with a gutter. Right graphic plots a non-visibility tunnel crossing with a reduction from two-way to a single lane.

(see left image in Fig. 1). However, there is no production system available to deal with these complex scenarios and just a few demonstrations have tackled this problem, recognizing its complexity [5]. A description of the perception-planning-control seamless integration to deal with these use cases is presented in the coming sections.

- **Narrow two-way road driving** represents a common scenario in rural areas. Highway lanes width are usually set at 3,5 meters in Europe with reductions up to 3m in urban areas to discourage high speeds in city centers [10]. When it comes to rural areas (see center image in Fig. 1), there are two main constrains to take into account: 1) standard vehicle width has increased in recent years from 1,8 to 2.2m (including rear-view mirror); and 2) lane width is reduced up to 2,5m in rural areas, increasing the complexity when crossing vehicles in two-way roads.

- **Tunnel driving** constitutes a technical challenge that has been already faced in urban areas. However, additional factors may increase the complexity in these scenarios in rural areas. Specifically, the vehicle crosses a tunnel that includes a lane reduction from two-way road up to a single narrow lane with a sharp turn when exiting (see right image in Fig. 1), adding priority traffic management and control capabilities to classical localization problems in these situations.

2.3 In-vehicle architecture

Autonomous driving is computationally intensive and requires integrating different functions such as perception, map service and localization, world model, navigation and decision-making, and vehicle control into a unified system that exhibits rational behavior. The arrangement and interaction of different functional components play a crucial role on the robustness and reliability of the vehicle operation.

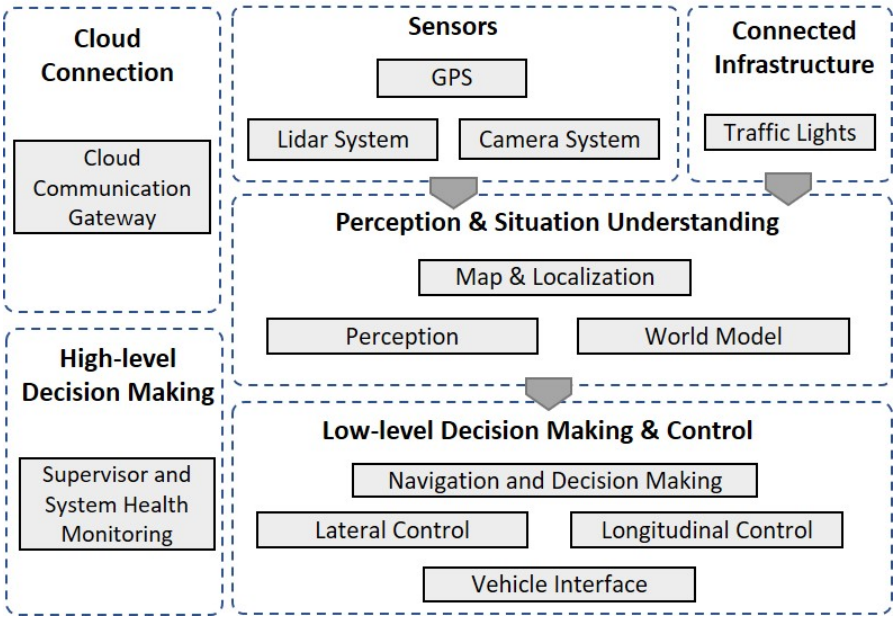


Figure 2: Functional components of Tornado Vehicle

Figure 2 shows different components of the vehicle. The following sections briefly explain each of these components:

Map and Localization: This component is responsible for gathering and centralizing all map related information, aiding to localize the vehicle within the available map. The map database stores the map of the operational area in a proprietary format that enables faster access. Communication with map service is implemented in query pattern. Requesting modules send a query to the map service component for receiving relevant information. The localization component provides position, heading, velocity and timing information, together with integrity information.

Cloud Support and Communications Links: This component provides a platform in which autonomous vehicles communicate with cloud server. The system mainly provides three functionalities: 1) Remote tracking and map visualization of all autonomous vehicles, providing them safety alerts and warnings such as, slippery zone or civil work in progress; 2) Remote control during critical situation like priority vehicle approaching; and 3) Connectivity between vehicles and cloud through cellular technology.

Supervisory System: This component monitors and controls all the components to ensure the safety and robustness of the operational modules. It selects the discrete behavior of the components, that is, to determine which behavior each of the functional components must have at each moment. It maintains different operational modes (e.g. autonomous, manual, standby) and sends signals to the appropriate operational components to switch their states accordingly.

Perception, *World Model*, *Navigation and Decision Making* and *Vehicle Control* components are detailed in the next sections.

2.4 Perception and World Model

This section details the environment detection and understanding in two subsections (perception and world model respectively) accordingly to the in-vehicle architecture.

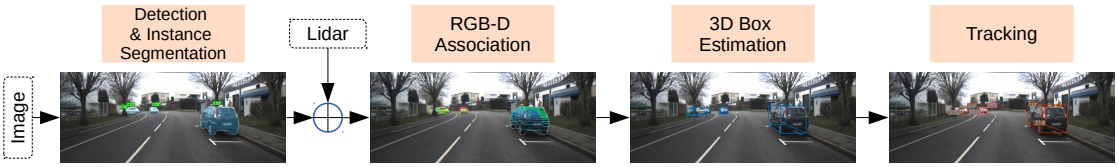


Figure 3: Pipeline for 3D object detection. Objects are detected in the images from the cameras. Later, LiDAR data is included to infer the geometrical properties of the objects. Finally, all the detections are merged, and a tracking stage adds temporal consistency across consecutive frames.

2.4.1 Perception

This section describes the sensor suite configuration and the algorithms designed to provide reliable detection of the obstacles in the surroundings of the vehicle. The system configuration is based on five evenly distributed cameras and a 32-layer LiDAR located in the center of the roof rack, providing a 360 degree field of view.

The information provided by the sensors is fused through a low-level fusion approach for obstacle detection, classification, 3D box estimation, and tracking. The system obtains vision-based detections from a robust state-of-the-art framework, and feed them to a 3D box estimation method that makes use of LiDAR information to infer their size and location. Finally, the tracking algorithm exploits this spatial reasoning to add time consistency and enhance the reliability of the final detections. An overview of the pipeline is depicted in Fig. 3.

Vision-based detection and classification: Detection and classification are based on computer vision approaches to take advantage of the feature-rich appearance information delivered by the cameras. The designed solution relies on the most recent and reliable detection algorithms, which provide superior performance over other alternatives. Furthermore, enhancing the detections with a pixel-wise semantic mask was proven useful to improve data association between both modalities (images and LiDAR data), provided that accurate extrinsic calibration is available. The Mask R-CNN framework [11] was adopted to perform both tasks jointly, as

it combines the high detection accuracy featured by the well-known Faster R-CNN [12] detector with the ability to perform instance segmentation.

Mask R-CNN is a two-stage method that, unlike other similar approaches, can achieve real-time framerates using deep convolutional networks. It accepts an RGB image as an input and feeds it through a set of stacked convolutional layers (backbone) responsible for extracting features that are shared for the subsequent tasks of detection, classification, and segmentation. Then, these features are used in the first stage to identify Regions of Interest (ROIs) in the image through an RPN (Region Proposal Network). Finally, in the second stage, features are pooled from these ROIs and propagated to successive dedicated layers to perform the final inference tasks. In this work, the ResNet-50 model [13] was adopted as the backbone because of its compelling performance and limited computational cost. It was also endowed with an FPN (Feature Pyramid Network) structure [14] to generate additional feature maps at different scales, allowing the identification of distant objects.

The outcome provided by this stage is a set of 2D bounding boxes with information about the category (e.g., car, person, etc.), each containing a pixel-wise mask that defines the contour of the object. Note that a different set of detections is obtained for each camera.

3D box estimation: Once the objects are identified within the images, LiDAR data is included in the pipeline to provide geometrical information. The high accuracy featured by LiDAR scanners in measuring distances justifies the use of this modality at this stage.

The semantic masks from Mask R-CNN allow identifying the LiDAR data (represented as 3D points) that belong to each obstacle. At this point, accurate extrinsic calibration becomes critical to ensure proper data association. In this case, an evolution of [15] suitable for monocular devices was used to obtain the extrinsic parameters that represent the relative position between sensors.

Although the semantic information from Mask R-CNN makes the assignment of LiDAR points to objects straightforward, it is still necessary to deal with the problem of pose and occlusions occasioned due to the nature of the sensors, i.e., information is limited to the visible surface of the objects. Meaningful representation of the environment should include, instead, information about the real geometry of the objects. In order to overcome this problem, the Frustum PointNet approach [16] was used to retrieve the geometrical structure behind the raw LiDAR representation of the obstacle. This approach is able to provide the size, location, and orientation of the obstacles using LiDAR information, and taking into account the missing parts. The use of the pixel-wise segmentation allows our approach to provide the 3D estimation stage with a more precise segmentation of the point cloud of every instance, obtaining better results. The expected outcome of this

stage is a set of positioned cuboids representing the real geometry and distance from the ego-vehicle to every obstacle.

LiDAR processing is performed as follows. In the first stage, outliers are removed through the use of an instance segmentation PointNet [17] that filters out the remaining noisy points that do not belong to the obstacle. Later, the 3D bounding box of the obstacle is computed through a two-step phase. The first step provides an estimate of the center of the obstacle through the T-Net network (based on the PointNet architecture). Once obtained, the points belonging to the obstacle are translated into this new reference. In the final step, a new PointNet-like network is used for the computation of the final oriented 3D box of the obstacles. As with the detection part, a different set of 3D boxes is obtained for each camera. At the end of this stage, all of them are merged through an NMS (non-maximum suppression) procedure that prevents eventual duplications.

Tracking: When the whole set of 3D detection is available and expressed in a common frame, the next step is to add consistency to the detection through the tracking stage. This stage allows to estimate and predict the movement of the agents in the scene. It is also able to handle situations where the obstacle is occluded for a limited time. The tracking algorithm used in the perception pipeline is composed of three stages. The first one is the movement estimation, based on an Unscented Kalman Filter [18] with different movement models for pedestrians and vehicles. The second stage is the ego-vehicle movement compensation, which is required to correct the misalignments on the subsequent detections due to the movement of the vehicle. This is done through the use of the GPS/INS system available in the vehicle. Finally, the last stage, data association, correlates the detection in the current cycle with the set of already tracked agents, adding new instances whenever it is necessary.

As a result of the tracking stage, the temporal coherence between consecutive frames can be exploited, avoiding inconsistencies and smoothing out any eventual deviation. As an example, Fig. 4 shows the output of the perception pipeline over time in a situation with three tracked agents.

2.4.2 World Model

The perception system provides information about the surrounding obstacles but without context information. By adding the digital map, *a priori* information can be added for scene understanding purposes (see Figure 2).

High Definition Vector Map: The operation area is described through a high definition vector map that was created specially for the experimentation. This map is a general representation of the road network and includes topological, geometrical and semantic information. It comprises details about the interactions that may exist between roads at a lane level detail. The center and the boundaries of each

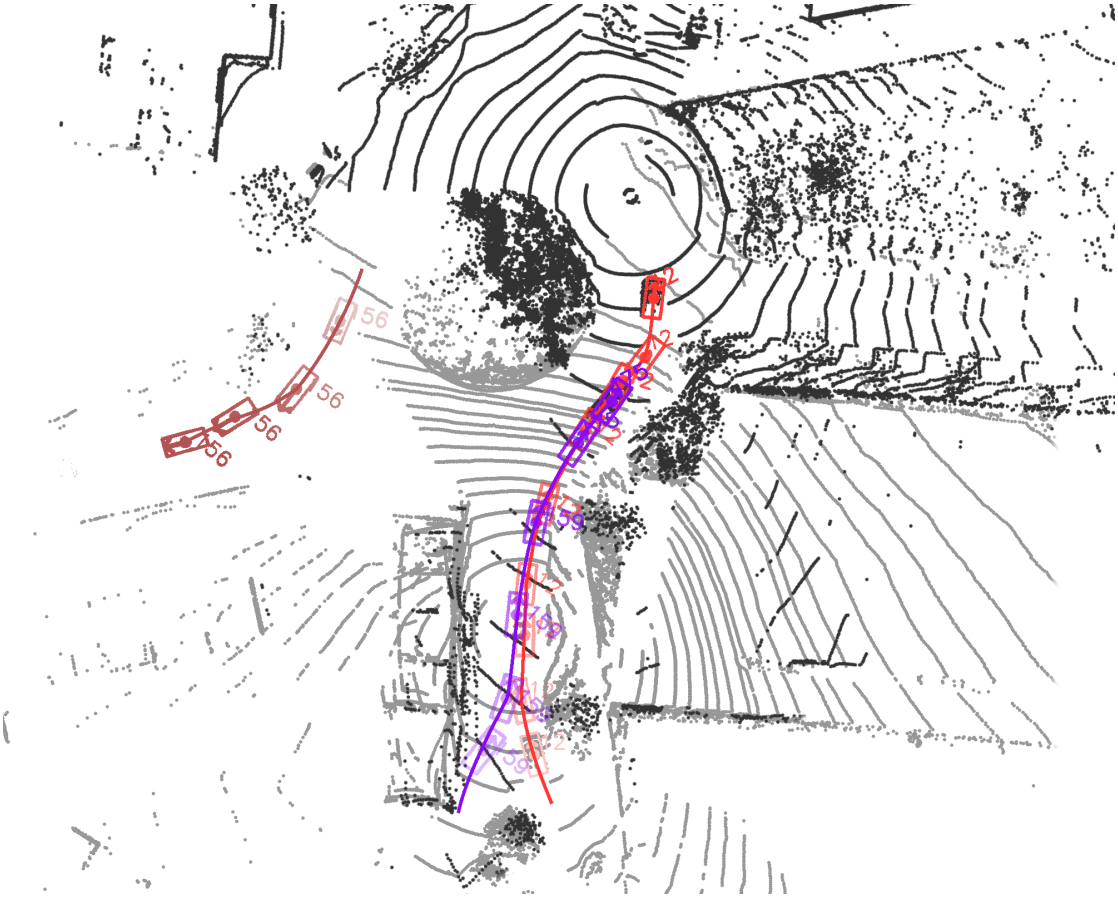


Figure 4: Bird's eye view representation of a roundabout scenario according to the information provided by the perception system. This example depicts three different obstacles and their trajectories over several frames, drawn over raw LiDAR data.

lane are also stored in the map with centimeter accuracy (for more details see Â§2.4.5 in [19]). In addition, semantics are associated with each lane as attributes representing information such as speed limit, marking types, driving directions, etc. In France, and especially in rural zones, markings and signs related to intersections may be challenging to perceive and understand. All information about intersection type, shapes or priority order is stored in the map. Traffic lights location and association to lanes leading to intersections are also stored in the map. All this knowledge is compiled as a geographical database within the vehicle. The Map Service system manages this database and makes it available to other vehicles' systems upon request (see Figure 2). The use of this map is threefold: 1) it allows computing the itinerary to reach the desired destination; 2) it supports the navigation module for real time computation of the trajectory while operating in autonomous mode; and 3) it provides context in which the vehicle is navigating.
Obstacles Contextualization: The map is used to contextualize the perception system output. This contextualization aims to filter the perceived objects by labelling them as *pertinent* or *not pertinent*. This is part of the situation understanding process as it is defined in [20]. For this purpose, pedestrians and vehicles are considered differently. All pedestrians are automatically labelled as *pertinent*, wherever they can be located with respect to the navigable space. On the contrary, vehicles and other obstacles are considered as *not pertinent* if they are not located on a space that is navigable by the ego vehicle.
Connected Traffic Lights: The experimental area includes a pair of connected traffic lights which regulates traffic in the tunnel where only one vehicle can navigate at a time. For this experimentation, the detection of traffic lights only relies on Vehicle to Infrastructure (V2I) communication as the embedded perception sensors do not return traffic lights information. The V2X devices use the 802.11p communication standard [21]. The state of the traffic lights is sent by the infrastructure through a Signal Phase and Timing (SPaT) standard message [22]. This message stores for each light information about its state, the time before state change, the ID, etc. When approaching an area equipped with V2I, the vehicle receives information about several traffic lights. After matching the perceived traffic lights with the map, it is straightforward to identify which traffic light is *pertinent* for the vehicle decision making and navigation.

2.5 Trajectory generation

The trajectory generation process takes into account a set of high level requirements related to regulatory rules, comfort and mission purposes. It aims at providing a safe and comfortable trajectory in real-time to the control while leading to the user-defined destination. The process is divided into two steps to limit computational effort, namely decision making in which the appropriate maneuver

is chosen; and trajectory planning in which an optimized short term trajectory is computed and transmitted to the control system. Moreover, the autonomous vehicle presented in this work aims at providing a mobility service within an operational area. When starting a new mission, a driving plan is computed to determine the shortest route to reach the demanded destination and evaluate the best a priori trajectory. Decision making and trajectory planning are then computed in real time based on this initial driving plan and detected obstacles.

2.5.1 Decision Making

The decision making process consists in evaluating the consequence of each obstacle and traffic light perceived at a given process time onto the initial driving plan. This driving plan is then modified according to the expected interaction between the ego vehicle and the detected obstacle in order to maintain a safe and comfortable behaviour. The expected interaction is formulated based on the relative obstacle position (e.g. same lane as ego vehicle, opposite lane, partially on ego lane, side walk), direction (i.e., moving along and backward or forward, moving perpendicular and to or away from the driving plan) and type (e.g. pedestrian, vehicle). The maneuver associated with each perceived obstacle is established in an expert system manner. By doing so, we keep computational complexity low and get a deterministic behaviour. Every maneuver adds constraints on the longitudinal (slow down, follow, stop) and lateral (avoid) dimensions which are taken into account in the trajectory planning process described in section 2.5.2.

Intersection management

Intersections require a special treatment because of their complexity. Having this in mind, an extra set of constraints is designed to handle intersection crossing. They consider not only the intersection detection thanks to the digital map but also their priority rules according to the interactions with other vehicles.

A lane-level description of the road network is provided by the map embedded in the vehicle. In case of intersecting or merging lanes, the map also provides the legal right-of-way between lanes. Finally, the type of intersection is also provided by the map (e.g. yield, stop). Every lane involved in the driving plan is studied by the decision making module to detect future intersection, applying the corresponding maneuver sequence. If a stop or yield intersection is detected along the path, an interest zone containing oncoming lanes is computed. The ego vehicle will therefore stop and wait if an obstacle is detected inside this interest zone.

Fig. 5 illustrates a sequence for roundabout management process. Fig. 5.a represents the approaching maneuver to the roundabout where vehicle speed is adapted. Once in the roundabout entrance, Fig. 5.b shows the interaction evalua-

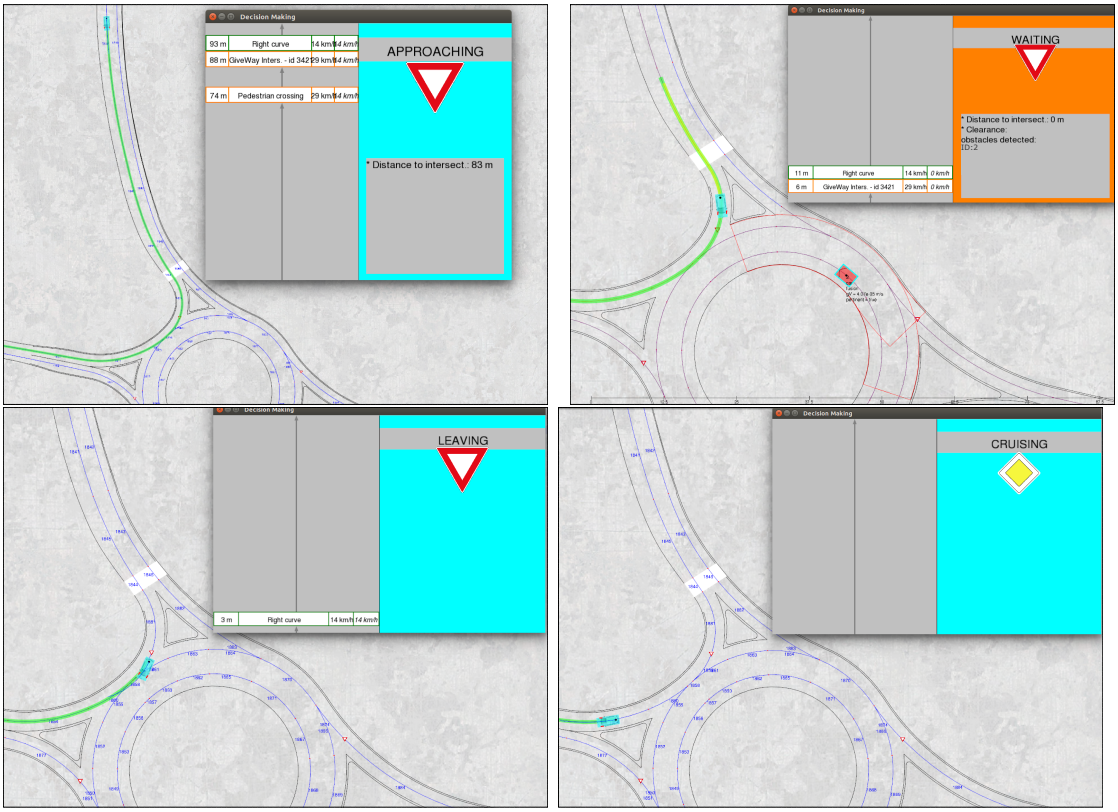


Figure 5: Intersection management, example of entering a roundabout. Ego vehicle is in blue and a detected vehicle is in red. The initial driving plan is represented in green and the area with right-of-way is formed by red polygons. The current status of decision making is shown in the top right window.

tion with other traffic agents to either continue or stop (as in the example). Once the vehicle is engaged (no traffic in the zone of interest), it enters the roundabout (see Fig. 5.c). Finally, Fig. 5.d represents the vehicle leaving the roundabout, re-adjusting vehicle speed according to future events.

This approach of decision making coupled with *a priori* driving plan provided encouraging results through the different scenarios. A transition state machine guarantees that no illegal decision is taken and that driving rules are respected. Moreover, there is no need to make an hypothesis on the number of obstacles to manage since each one is taken independently and provides constraints to the path planning algorithm. The decision making process takes into account the perception system capabilities to avoid mistaken decisions. This allows to account for mistaken contextualization of obstacles, position and speed measurement errors, among others, pushing the vehicle capabilities to its limits.

2.5.2 Trajectory planning

The short-term trajectory of the ego vehicle is computed in real time, including the geometric description of the desired path, lateral error tolerance and associated speed profile.

Space definition

High definition vector map provides a geometric description of each lane which is used to create lateral boundaries (i.e., a driving corridor) limits to the vehicle. A longitudinal boundary is set with respect to the closest obstacle for which the decision system chooses a stop maneuver. Spatio-temporal footprint of each obstacle associated with an avoidance maneuver is computed based on estimated position and speed. The available driving space is therefore reduced according to the set of perceived obstacles.

Path optimization

This component is in charge of optimizing the driving space in the short term horizon. The driving space is sampled into a set of vertices along regularly spaced transverse lines. They are candidates for constructing the path using a set of connected splines as illustrated by Fig. 6. Splines combinations are convenient to compute a path complying with second order geometric continuity [23]. Moreover, closed form expressions of each circle spline can be found which makes efficient the final sampling of the resulting path.

The full circle spline combination set is computed in order to create a graph of possible solutions for the current computation step. This table demands high computational resources. However, it happens only at the first time step since it is simply updated while the vehicle progresses on its trajectory.

Finally the optimal circle spline compilation is searched using a Dijkstra algorithm [24]. The cost function is composed of three terms in order to take into account:

- Distance to driving space boundaries. High costs are associated with candidates that make the vehicle getting close to the boundaries. The physical vehicle's dimensions are taken into account.
- Distance to initial driving plan. Cost increases with area between candidate and initial driving plan.
- Derivative similarity. Cost increases with the difference between candidate's derivative and initial driving plan derivative. This provides high cost to oscillating candidates.

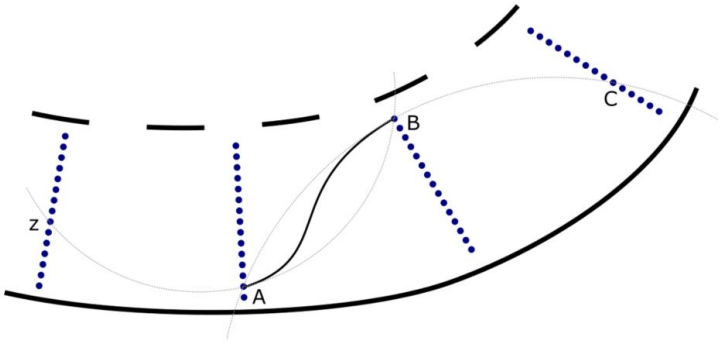


Figure 6: Sampling of driving space with vertices (blue dots) arranged in layers. An example of a circle spline that connects second and third layer is shown in solid black. Continuity with first and fourth layer are shown in dotted black lines.

Dijkstra search algorithm explores in priority the less cost solution. This results in fast solution convergence if the initial driving plan is still convenient at the current time step. However, more candidates are evaluated if current constraints make the initial driving plan infeasible which increases significantly the computation time. If no solution is found, the previous solution is kept but the associated speed profile sharply falls to make the vehicle brake promptly.

Speed profile

The final step of trajectory planning consists in associating a speed profile to the optimal path. Initial driving plan provides a first skeleton on which maneuvers associated with obstacles apply extra constraints. The final speed profile is obtained by filtering and smoothing them as illustrated by Fig. 7. The consolidated trajectory is finally sent to the control system as a set of waypoints with information about vehicle position, heading, speed, acceleration and road curvature within a speed-based horizon.

2.6 Vehicle dynamic control

This section describes both lateral and longitudinal controllers. They use as main inputs the positioning and navigation systems' information.

2.6.1 Lateral control

First Tornado project demonstration in 2018 [25] proposed two different lateral controllers: 1) a path-tracker controller able to provide a zero tracking error, increasing the safety feeling on-board; and 2) a predicted controller minimizing the control effort whereas keeping the vehicle in the lane (mimicking human behavior).

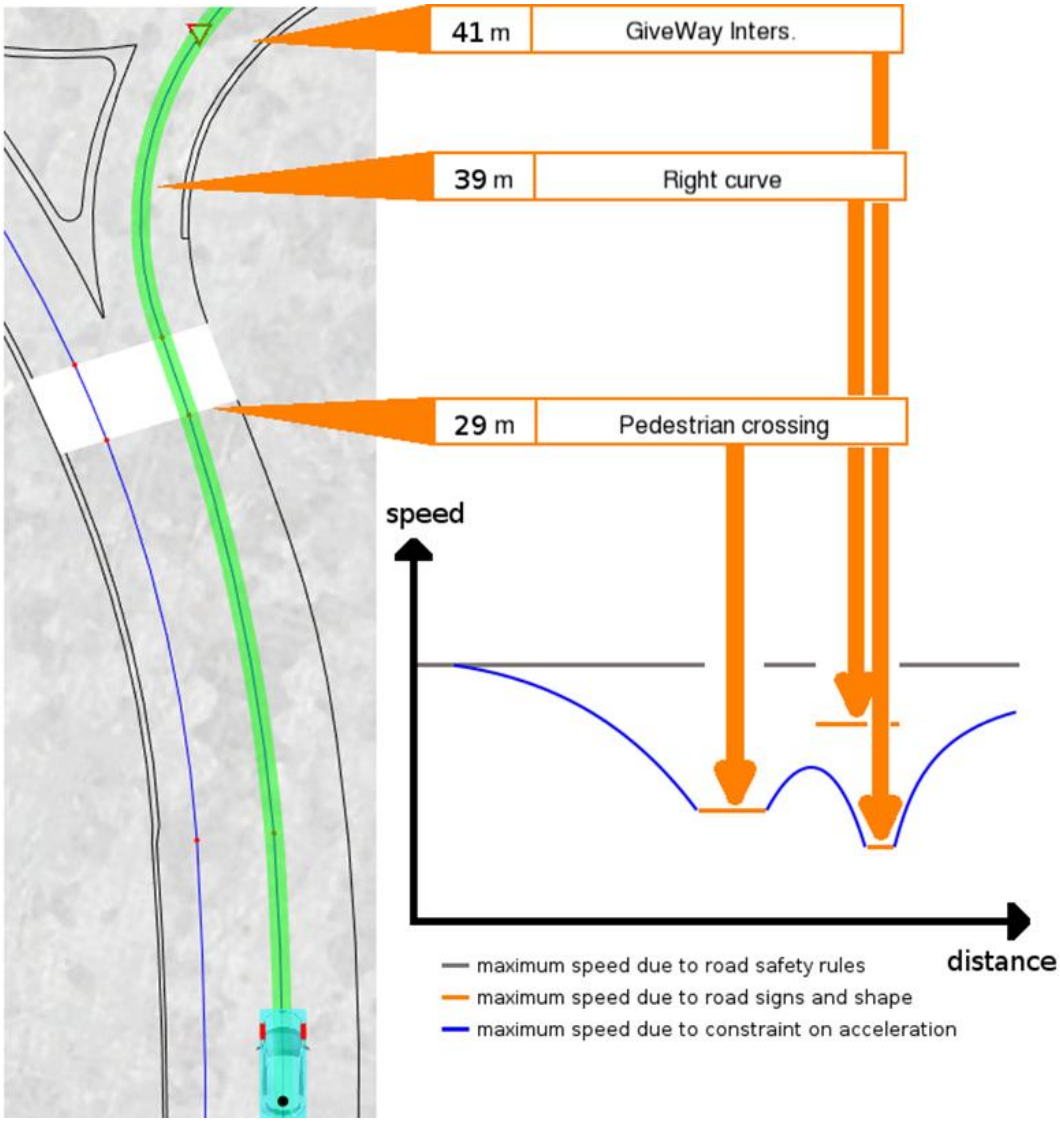


Figure 7: Speed profile computation

Both lateral controllers are based on a look-ahead dependent minimization of the desired yaw rate [26]. The controller is a second order transfer function with a gain K, and a variable look-ahead distance d. Its objective is to determine the steering wheel angle δ_c according to a yaw rate error w_r calculated as the difference between desired w_d and measured w_v yaw rates. The desired yaw rate w_d is equivalent to the vehicle longitudinal speed v divided by the road curvature ρ in the target point (set at a look-ahead distance d). Notice how K and d were tuned in [25] to achieve different behavior for the tracker K_1 and the predicted controllers K_2. K_1 controller was optimized to perfectly follow the navigation path, providing a safety feeling to the passenger in terms of precision. However, it exhibited an aggressive behavior in roundabouts and lane changes. The main reason behind this undesirable behavior was that the path tracker controller was tuned to be a fast controller. On the contrary, K_2 was a relaxed version of the path tracker controller, taking advantage of the driving corridor provided by the navigation system. It was optimized to be slower, less precise, but exhibiting less overshoot in lane changes and roundabouts. The predicted controller was more comfortable but the tracking feeling was downgraded with respect to K_1.

Both controllers are here implemented in a single control structure based on users' feedback during the 2018 demonstration (see [25] for details). This control structure is based on the Youla-Kucera parametrization [27].

Vehicle lateral model G and both controllers K_1 and K_2 are factorized as the product of stable left and right coprime transfer functions (see [28] for more details) :

$$\begin{aligned} G &= NM^{-1} = \tilde{M}^{-1}\tilde{N} \\ K_1 &= UV^{-1} = \tilde{V}^{-1}\tilde{U} \\ K_2 &= U'V'^{-1} = \tilde{V'}^{-1}\tilde{U'} \end{aligned} \tag{1}$$

The Youla-Kucera parameter Q permits to describe the class of all stabilizing controllers; but it is here used for connection of controllers K_1 and K_2: $Q = \tilde{V'}(K_2 - K_1)V$. The final control structure is expressed as follows:

$$K(\gamma Q) = (U + \gamma QM)(V + \gamma QN)^{-1} \tag{2}$$

where $\gamma \in [0, 1]$ is computed online with respect to the corridor width provided by the navigation system and the lateral error. Different values of γ activate different level of controllers K_1 and K_2 ($\gamma = 0$ is K_1, $\gamma = 1$ is K_2, and $\gamma \in [0, 1]$ is a mixed behavior between both controllers). Fig. 8 clarifies how the γ-based decision-making system works together with the Youla-Kucera based control structure. The supervisor module checks first the corridor width, and then the lateral error, providing the value of γ that activates the corresponding controller:

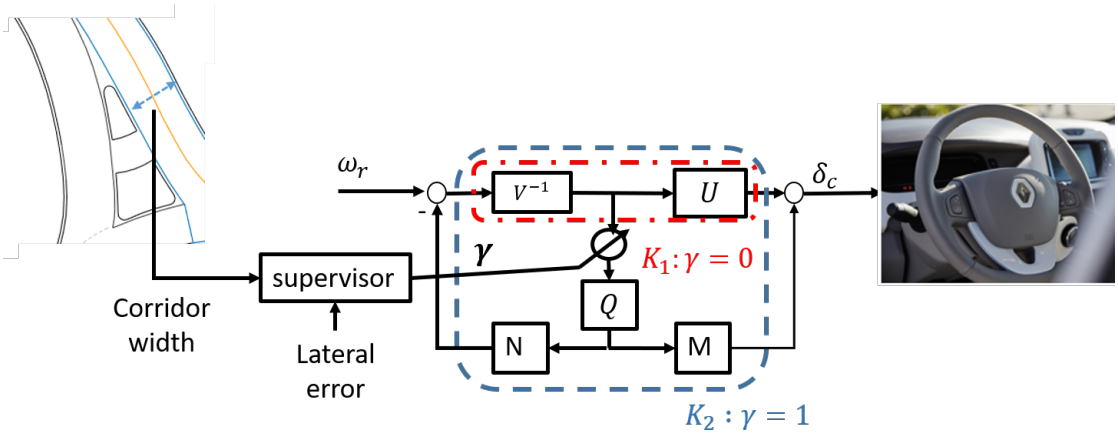


Figure 8: Vehicle lateral controller

- In the roundabout entrance and exit, the predicted controller is activated since the driving corridor is wide, allowing significant lateral errors. This leads to smoother control efforts, having a more comfortable control response.
- In the narrow two-way road and the tunnel, the tracker controller is activated. This is caused by the reduction of the driving corridor, demanding a higher tracking precision.
- In the other areas γ changes gradually between $[0, 1]$ according to the current lateral error [29]. It activates the adequate proportion of each controller and performs a smooth transition between the two previous areas.

2.6.2 Longitudinal control

The experimental platform is already equipped with a low-level longitudinal controller. This low-level controller receives speed reference from the navigation system, actuating accordingly over throttle and brake pedals. Reference/output behavior are in blue and red solid lines respectively in Fig. 10. Notice how discontinuities appear in the speed reference (around second 22) and how the low-level controller takes around 400 ms to reach the desired speed.

To deal with such discontinuities, a high level longitudinal controller with better tracking capabilities and robustness was designed. A Linear Parameter Varying (LPV) controller is proposed, having as scheduling parameter the speed reference difference from one sample to the other. Specifically, the low-level controller behavior is considered as an open-loop system, together with a LPV weighting function based on the desired sensitivity of the closed-loop (CL) system. Sensitivity transfer function bandwidth changes with the scheduling parameter, allowing a faster

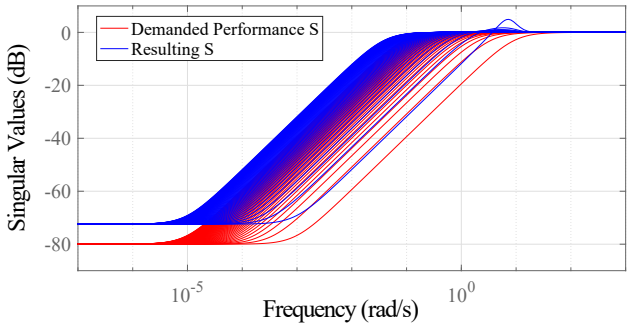


Figure 9: Sensitivity transfer functions for high level LPV longitudinal controller.

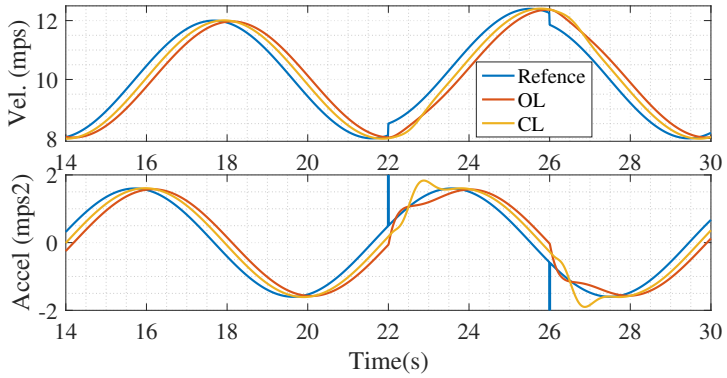


Figure 10: Longitudinal dynamics comparison. Longitudinal low-level controller (OL) and designed high-level LPV controller (CL).

tracking capability when the speed reference change is small (less than $\pm 2\ m/s^2$), and slowing down the response when the difference is greater. A gridding-based LPV synthesis approach [30] [31] is used for solving corresponding inequalities in a specific scheduling parameter range. Reference and resulting sensitivities are in red and blue in Fig. 9 for the whole scheduling parameter range. Resulting LPV controller behavior is in solid yellow line in Fig. 10. The designed high level controller permits a faster response (200 ms) with better tracking capabilities, filtering out discontinuities in navigation speed reference, and maintaining a comfort acceleration in the desired range ($\pm 2\ m/s^2$) .

2.7 Experimental results

This section describes the experimental tests conducted during two weeks at Rambouillet. A Renault ZOE vehicle travelled 7.5km per itinerary. Along these kilometers, three scenarios were the main technical challenges in the demonstration:

A roundabout, a narrow two-way road and a tunnel (see Fig. 1 for details). Fig. 11 presents a vehicle performance zoom-in for the three specific scenarios. Fig. 12 shows experimental results on the full itinerary whereas orange, pink, blue and grey zones represent the roundabout driving, the narrow S-shaped, the tunnel crossing and the U-turn (in manual mode) respectively. The top graph shows the vehicle lateral error. The next one depicts the vehicle steering wheel angle. The following graph points to the steering wheel angle rate. The bottom graph shows the speed reference from the navigation (solid red lane) and the current vehicle speed (solid blue line).

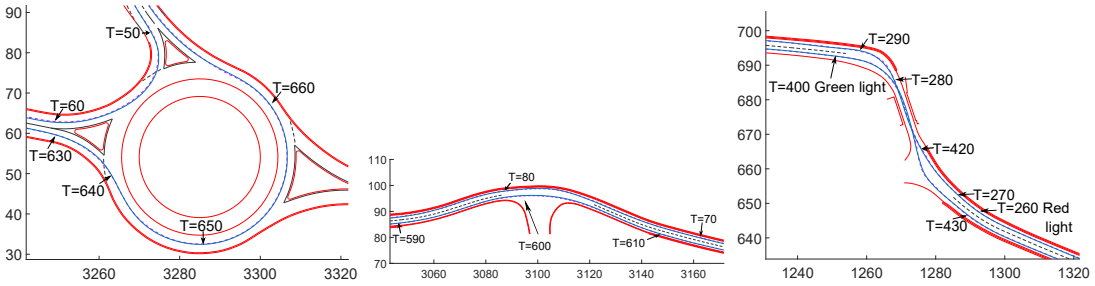


Figure 11: The planned trajectory is plotted as a dotted magenta line, the vehicle effective path is shown in blue, boundaries are depicted in red, and road lines are plotted in black. (a) Vehicle performance in a roundabout. (b) Vehicle performance in a narrow “S-shaped" road section. (c) Vehicle performance in the tunnel scenario.

2.7.1 Scenarii analysis

Results related to the roundabout scenario are depicted in Fig **??**. The vehicle crosses this area twice: first between seconds 50 and 60 while entering the north roundabout access and driving right to the west exit; and then between seconds 630 and 670, where the vehicle enters the roundabout through the west access and goes north to continue its way back. The performance of the vehicle is visibly good, accurately tracking the planned trajectory (see lateral error in the orange zone on the top graph of Fig. 12), adjusting its speed (second 50) or even stopping when others vehicle in the roundabout (second 630) when approaching the roundabout entrance (see bottom graph of Fig. 12). The steering angle and the steering wheel rate are kept within the vehicle's limits.

The two-way narrow road driving is shown in a S-shaped section (see Fig **??**). It depicts one of the most difficult scenarios with little tracking error margin. The vehicle travels through this section twice: 1) Between seconds 70 and 90 from right to left in the figure, where the tire is always close to a gutter on the right side; and 2) Between seconds 590 and 610 where a curb is present on the right. The tracking performance can be seen in the top graph of Fig. 12 in the

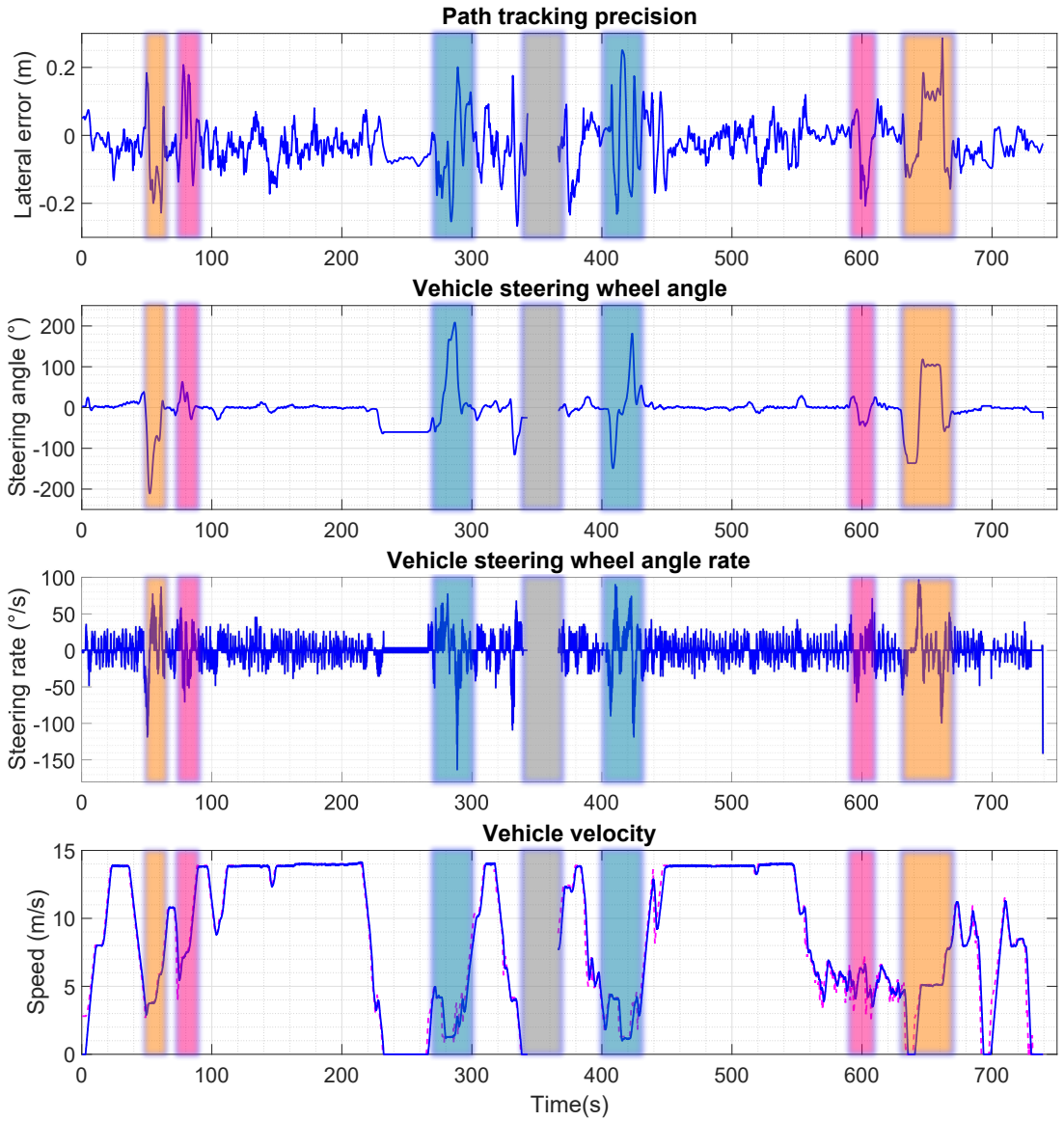


Figure 12: Vehicle performance for the round trip proposed experience.

magenta areas. The vehicle lateral error never goes larger than 0.2 m precision, ensuring the proper behavior even when turning and maneuvering in narrow two-way roads, keeping steering angle and steering rate vehicle response within vehicle boundaries (see second and third graphs from the top in Fig. 12 respectively). The speed profile (see bottom graph in Fig. 12) indicates how the vehicle adapts its speed before reaching the S-shaped (second 70), decoupling lateral and longitudinal accelerations as requested by users during 2018 demonstration [25]. Interestingly, the vehicle is following a bicycle (second 600 s) in the way back, modifying in

real-time the speed profile accordingly.

Fig. 35 presents a bird view of the tunnel scenario. The vehicle also crosses this area twice: first between seconds 260 to 300; and then between seconds 400 to 430. The scenario depicts a one way tunnel where crossing priority is assigned via two coordinated V2I traffic lights at each side of the tunnel. This is one of the most challenging scenarios for the vehicle since it has to first manage the traffic light, then passes through a narrow tunnel, reducing its speed since the positioning system is degraded. This is visible in the bottom plot in Fig. 12 between seconds 280 and 420, when the speed is significantly reduced whereas the positioning system is recovering its precision, autonomously driving in a degraded mode. The top graph in Fig. 12 shows tracking error (including positioning system inaccuracy), where the vehicle exhibits a good performance, keeping the vehicle in the lane. During this time, K_2 controller is fully activated to provide smooth steering wheel changes (see second and third graph in in Fig. 12), increasing the comfort on-board. Time between seconds 230 and 260 when the vehicle has zero velocity represents the moment when it's stopped in front of the traffic light. The steering angle remains static at the desired angle and the steering rate goes to zero until traffic light is green and the vehicle goes through the tunnel. The steering signal is continuous, giving a smooth and comfortable vehicle behavior even in full stop situations.

2.7.2 Overall performance

Performance out of the challenging zones (see Fig. 12 for details) exhibits also good results and confirms a proper behavior of the vehicle throughout the itinerary. The top graph of Fig. 12 shows that the lateral error is always lower than 0.1 m when out of the painted areas. The steering wheel angle rate (see third graph in Fig. 12) is kept under 200 degrees (maximum system capabilities) even in challenging conditions as roundabouts or tunnel scenarios. The speed shows a proper tracking (in blue) of the speed trajectory (in red), even when following cyclists (see between seconds 560 and 640 in the bottom part of Fig. 12.

Finally, Fig. 13 presents the acceleration GG diagram. The accelerations were limited to 0.2 g for the lateral axis of the vehicle, 0.2 g for the longitudinal acceleration and 0.25 g for braking accordingly to users' feedback in 2018 demonstration [25]. The correspondent acceleration ellipse is depicted in orange in the diagram. The acceleration profile of the vehicle is presented in blue and the acceleration points out of the ellipse are presented in magenta. Interestingly, the vehicle acceleration points are 99.25% inside the circle, thus respecting the acceleration constraints, showing that the ride is comfortable and safe at all times.

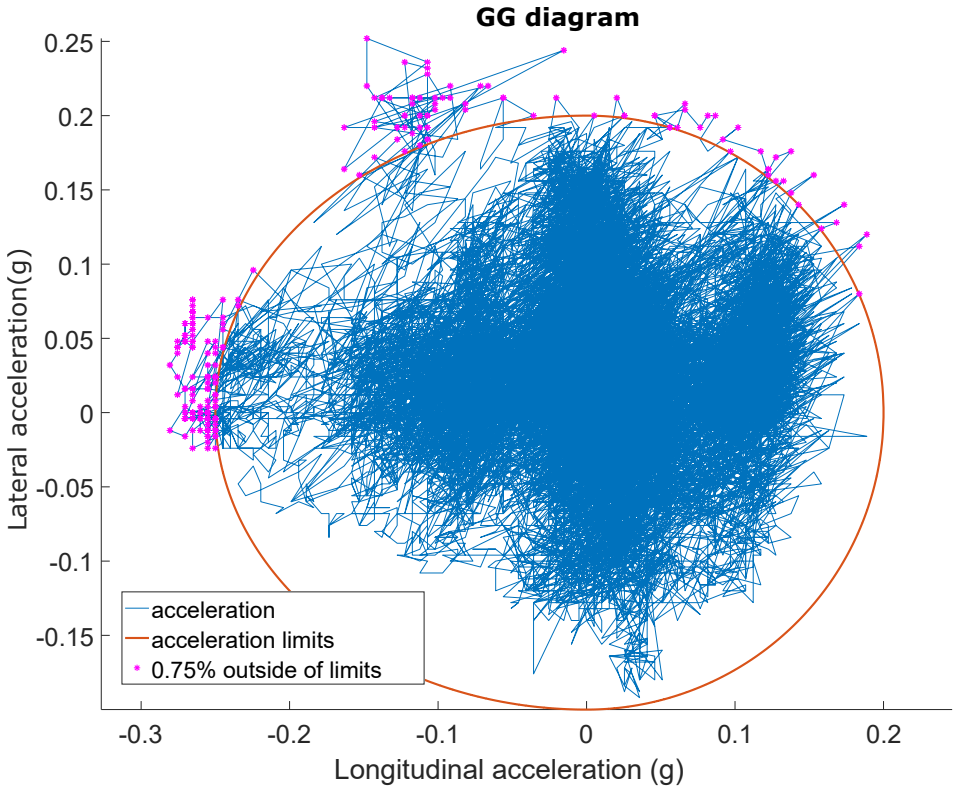


Figure 13: Acceleration GG diagram throughout the itinerary. The orange circle depicts the acceleration limits for the demonstration.

2.7.3 Processing time analysis

One of the challenges in such implementations lies in the capacity for the autonomous vehicle to react correctly to real traffic conditions. Despite their complexity, algorithms involved in autonomous driving shall run with reasonably low cycle times. This section provides some insights about the two main processing time consumers, namely Perception system and Navigation and Decision Making system (see Fig. 2).

Perception

The configuration of the perception pipeline was focused on efficiency, targeting rates above 10 FPS. The most computationally intensive processes (i.e., object detection and 3D box estimation) are carried out in parallel for all the cameras. Tests in a sequence fragment (shown in Fig. 14) yield an average run time of 63.0 ms (with 6.8 ms of standard deviation) for the detection stage and 21.3 ms (with 6.8 ms of standard deviation) for the 3D box estimation part. The run time of the RGB-D association step, which is in between the previous two, is almost negligible,

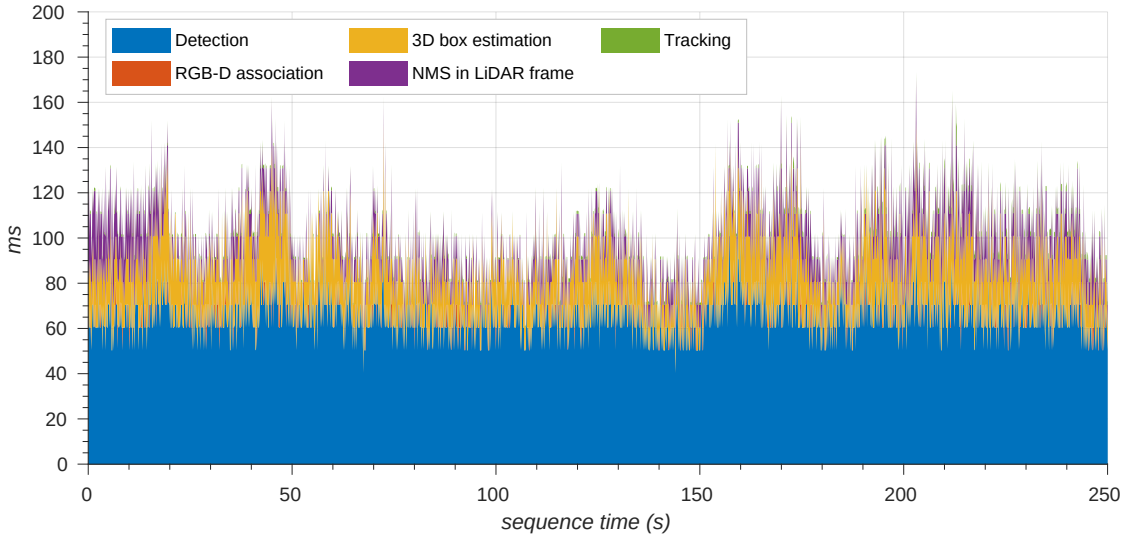


Figure 14: Breakdown of computation times in the perception pipeline during a sequence fragment.

as is the case with the final tracking stage (they take around 1.5 ms each). The latter requires, however, a previous procedure to express the detections from all the cameras in the LiDAR reference frame and perform NMS that takes 6.9 ms (3.3 ms of standard deviation).

Due to the design of the inference frameworks, these run times depend on the number of objects found in the environment, although this dependency is reasonably limited (as evidenced by the low standard deviations). Values reported above correspond to a relatively crowded scenario with 5.7 obstacles per camera; in the extreme case where 24 agents are found per camera, the detection stage reaches a maximum of 84.9 ms, and the 3D box estimation one, of 45.7 ms. As shown in the breakdown in Fig. 14, the sum of times in a cycle is often above 100 ms; however, several frames can be processed at the same time in different parts of the pipeline so that a new cycle can start before the end of the previous one. Therefore, the perception pipeline can sustain average rates of around 10 Hz.

Navigation and Decision Making

As introduced in section 2.5, real time operations of this system is composed of two main steps: Decision Making and Trajectory planning. Fig. 15 presents the computation time breakdown at each cycle for a complete user journey. The smallest part of the time budget is the decision making process since it takes an average time of 2.8 ms (with 0.92 ms standard deviation).

The second time consumer is for the initial step of trajectory planning: space definition. This takes an average time of 16.5 ms (with 6.3 ms of standard de-

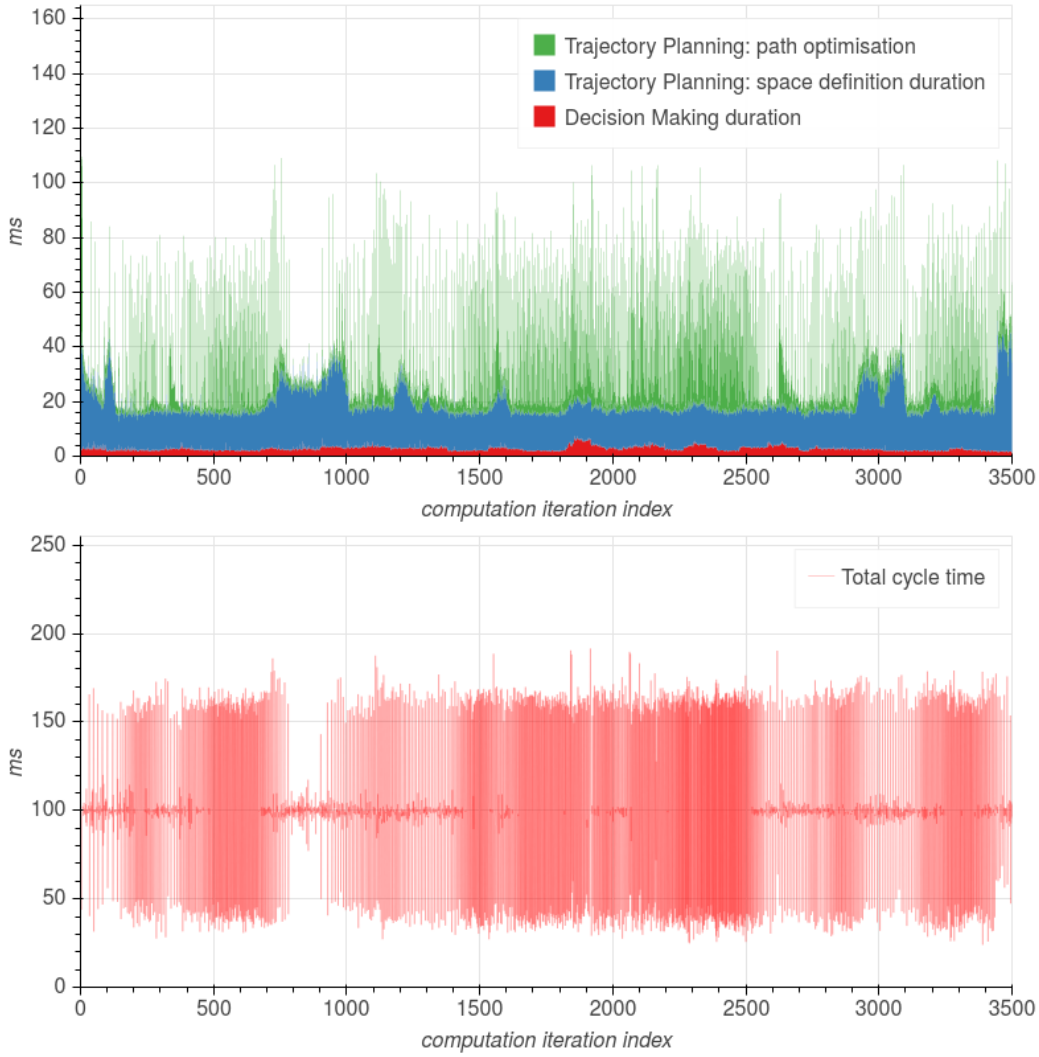


Figure 15: Navigation and Decision Making computation time during autonomous drive.

viation). This process consists in geometric construction of the navigable space so a large number of operations is required. Moreover, when interactions with other vehicles occur (particularly for iterations 700 to 1000 and 2900 to 3100), the process is more complex so computation time increases.

Then, path optimisation takes a similar average time than space definitions (17.9 ms) but has larger variability with a standard deviation of 26.1 ms. It can be seen by the multiple green spikes in Fig. 15. The computation complexity significantly increases when the vehicle has driven a sufficient distance to create a new vertex layer and all the associated candidate splines (see Fig. 6). These spikes are therefore more rare when the vehicle drives at low speed (e.g. from iteration 700 to 1000).

Finally, the complete trajectory is sent to Control system with an average period of 100 ms but with a quite high variability (41.3 ms standard deviation) due to path optimisation as shown in Fig. **??**.

2.8 Lessons learned

The main idea behind the trials and the two-week demonstration with different passengers on-board was to evaluate the need for novel mobility systems in peri-urban and rural areas. Interestingly, people found the system good enough to either replace or cohabit with current transport systems but there were still some remarks on the way the vehicle negotiate some specific interactions. The environment understanding and the decision associated with it remain as the two main challenges to technically overcome in the near future.

Driving rules violation constitutes one of the main blocking points for the system deployment. As example, some drivers did not respect the traffic light installed in the tunnel to deal with visibility problems (one way tunnel and sharp turn when exiting). Even if during informal talks with Rambouillet residents they pointed out that the traffic light in the tunnel was helping to remove an old road safety problem; it created some critical situations with the automated prototype (i.e., both vehicles at the same time in the middle of the tunnel) from people that crossed in red. This kind of situation that for two human-driven vehicles are easy to negotiate demonstrates the complexity when it comes to putting autonomous vehicles on the road.

A clear improvement point (also mentioned in previous autonomous driving demonstrations [5,6]) was the ability to properly manage roundabouts. Especially in rural areas, two-lane roundabouts allow cutting both lanes, describing almost a straight line for drivers that are used to the roads. This means that a considerably longer detection range is required to handle such drivers, pointing out to the adoption of V2X technologies in those situations to improve vehicle performance towards a more natural response.

Finally, it is worth to mention that perception system was conceived for urban environments (i.e. high interaction with other road agents and moderate speeds). This is clearly linked to the Operation Design Domain (ODD) concept, meaning that the specific sensor set-up should be enlarged to deal with higher speeds (i.e. highway driving).

2.9 Conclusions

The development and deployment of automated transport systems in low-dense areas represent a realistic application domain for autonomous vehicle: less interaction with other road agents (i.e., vehicles, pedestrians or bikes); limited and

repetitive driving areas that significantly help in terms of having an up-to-date digital map with high precision; and a realistic use-case where the autonomous vehicle can replace a classical public transport system or even create it with less constrains.

Next steps will be focused on removing the current technical barriers (i.e. the ability to deal with unexpected circumstances as other vehicles passing the traffic light in red) as well as enlarging the testing area towards a more complete mobility service. These objectives match with users' expectations collected via a questionnaire after experience the ride, demanding even a longer itinerary with more connections.

3 Inria

3.1 Introduction

The TORNADO project is a multi-partner project, subsidized by the Fonds Unique Interministeriel (FUI), which aims to study the interactions of the autonomous vehicle and the infrastructure for mobility services in low density areas and to offer new mobility services.

Within this project, under Lot 4, our work developed focuses on navigation of autonomous vehicle on Route du Bray, Rambouillet. Route du Bray is a two-lane, two-way country side road, situated in low-density area. While there is little traffic, vehicle navigation on this road brings with itself its own challenges. The lanes are narrow with occasional faded lane markings and contains numerous bumps where road repairs have been made in the past. The traffic speed limit goes up to 70km/h and in case of oncoming traffic, both the vehicles must displace slightly out of the lane to maintain a safe lateral gap between them. Due to numerous constraints, it is crucial that an autonomous vehicle is able to perceive the road curves or other vehicles on time and follow traffic rules and social behavior.

Figure 16: Route du Bray illustrated on Open Street Map.

To address the problem of vehicle navigation, we have built a system architecture that deals with various segments of autonomous driving including hierarchical decision-making, vehicle localization, perception of the environment, V2X communication, path planning and control of the vehicle. Figure 17 shows the key components of the system in block diagram. We split the system architecture in three main components:

- Knowledge base
- Hierarchical decision-making
- V2X communication

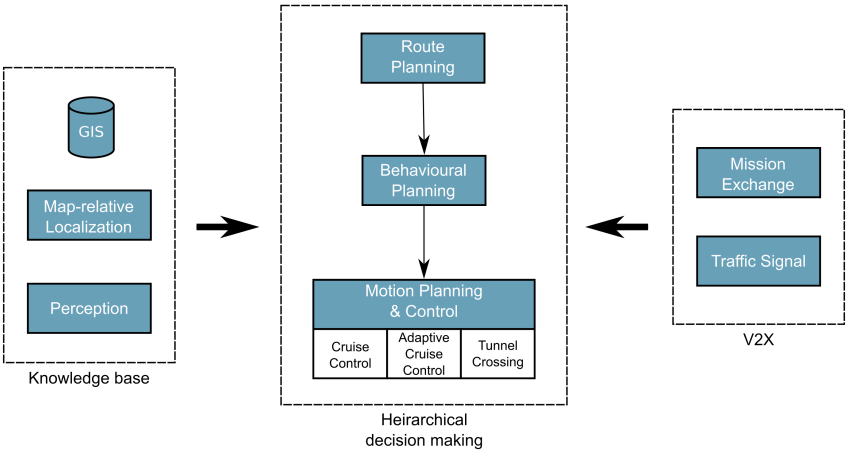


Figure 17: System architecture for autonomous driving on Route du Bray.

Decision-making requires reliable and adequate representation of the surrounding environment and traffic scene. For this, prior knowledge and sensory information must be processed so that it can be used for scene understanding and reasoning. Knowledge base prepares the necessary information based on 3 sources:

1. Offline map
2. Localization of the vehicle on the map
3. Environmental perception

Hierarchical decision-making deals with the given task of navigating the vehicle from one point to another. It is commonly split into three modules: route, behavioral and motion planning. Route planner deals with high level navigation task of finding the path to the final destination, while the behavioural planner is responsible for manoeuvre decision-making and ensures that vehicle follows required road rules and interacts with other road obstacles accordingly. Based on the selected behaviour, corresponding motion planner translates the semantic information to numeric implementation and generates appropriate paths and sets of actions. Motion planning is closely connected to manoeuvre execution and drive control and are thus presented together.

Additional to knowledge base and decision-making, an important part of the project is V2X (Vehicle-to-everything) communication. Modules have been developed for mission exchange with the long distance shuttle service and the reception of traffic signal status involved in the decision making of crossing the tunnel.

In the following chapters, we discuss the developed modules in detail.

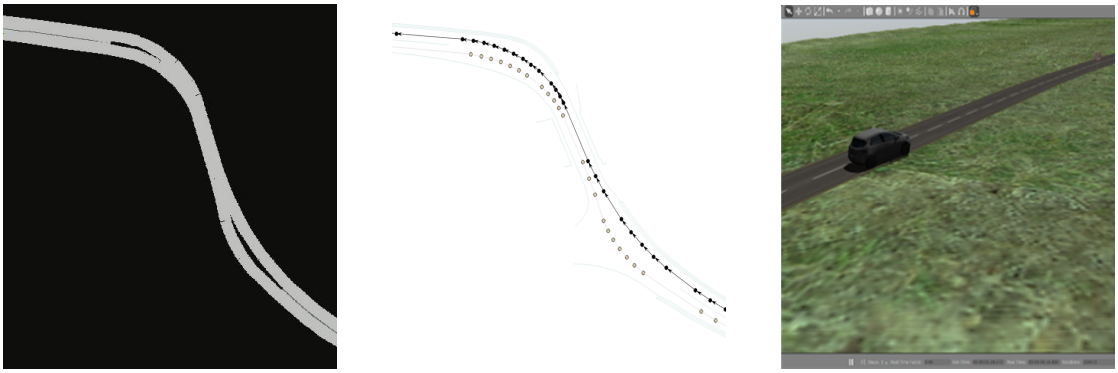
3.2 Knowledge Base

3.2.1 Offline map

An offline map database, provided by Renault, is utilized as prior knowledge of the road network. The database provides layouts of the drivable lanes, lane markings and road signs defined in WGS84(latitude/longitude) coordinates. Using the map database, metric and topological maps of Route du Bray are generated. Both metric and topological maps play significant role in representation of traffic scene as input for decision-making.

For the metric map, the map information is translated into a 2D grid representation as map grid and WGS84 coordinates are converted to a local ENU coordinates system. To incorporate a local navigation frame for Route du Bray, a local tangent plane is defined using East, North, Up (ENU) coordinate frame with its origin defined at arbitrarily chosen fixed coordinates. This is acceptable for a small map within a radius of few kilometers with the assumption that the change in earth curvature is negligible. By convention x-axis of the frame points to the East, y-axis to the North and z-axis is oriented upwards with respect to WGS84 ellipsoid.

Route du Bray is divided into 8 overlapping 2D grid maps. Figure. 18a shows a small segment of the map grid around the tunnel. The map grid has a resolution of 10 cm and represents 3 spaces: road in white, central lane marking in gray and no road in black.



(a) (b) (c)

Figure 18: Offline map. a) metric grid map, b) topological map, c) Gazebo simulation.

For the topological map, lane-level waypoints available in database are used to define a topological graph comprising of nodes and directional edges. Nodes consist of position information defined in WGS84 coordinates in the center of lane while the edges comprise of two connected nodes and additional necessary information such as lane width, lane position on the road, and characteristics of road segment

itself. The weight of each edge is set to its respective length, i.e. the distance between its nodes. Figure.18b illustrates topological graph at the tunnel in the direction of Gare de Gazeran. This topological map not only aids in localization and perception but also plays a key role in route and behavioral planning.

Availability of offline map also enabled us to build the simulation environment, Fig. 18c. Gazebo, an open source simulator, is used to test developed algorithms. Simulation environment of Route du Bray is prepared that contains road, lane boundaries and road signal information available in the database.

3.2.2 Localization

For situational awareness, localization of the vehicle within the map is very important. It is common in the literature to fuse different types of sensors to improve the vehicles localization. Fusing an offline map with on-board vehicle sensors provides complementary benefits. An accurate map helps integrate acquired sensor data and also provides information that is outside of sensors reach.

The map relative localization approach developed is based on Extended Kalman filtering (EKF), as illustrated in Fig. 19. The multisensor EKF fuses GPS and dead reckoning comprising of INS feedback and wheel odometry, as well as two distinct map matching algorithms: i) Iterative Closest Point (ICP) based visual lane level map matching is performed with visual lane tracker and grid map ii) decision-rule based approach is used to perform topological map matching.

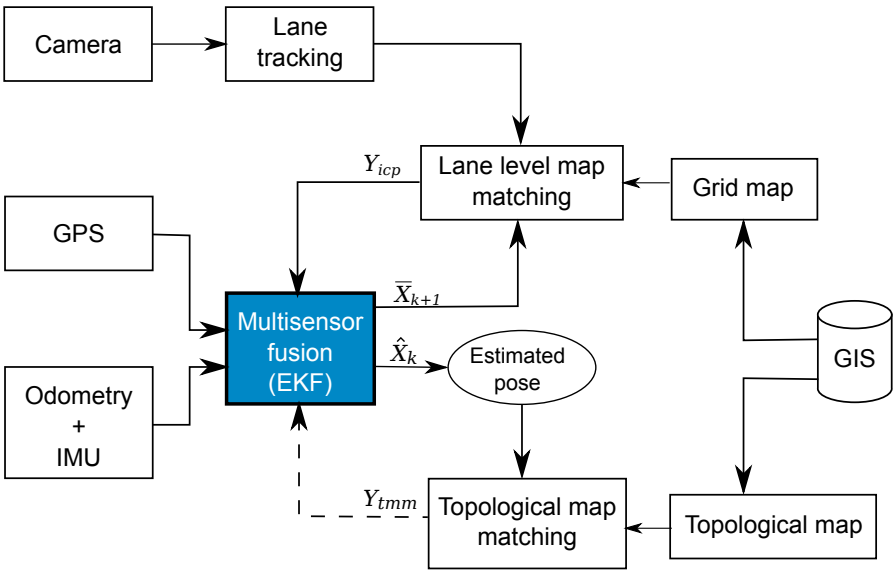


Figure 19: Overview of the map relative localization approach.

The state vector X_k consists of vehicle's pose, twist and linear acceleration in

three-dimensional space. Vehicle's kinematic model is defined by classical unicycle model and the pose of the vehicle in 2D is represented by vehicle's position (x, y) and it's heading θ. GPS measurements are converted to ENU coordinates in local navigation frame (discussed previously in section 3.2.1) and the position of the vehicle in 2D is used for measurement update. GPS observation is only made when the vehicle is in motion and GPS error is used to define its covariance matrix. The map matching algorithms and their fusion with EKF are presented. The details of lane markings and topological map matching based localization shown in Fig. 19 are discussed in the sections 3.2.2 and 3.2.2 respectively.

Lane-level map matching For visual lane marking based localization, metric grid map and camera based lane tracker are used. A particle filter based lane tracker [32] is implemented, that takes camera image and camera's intrinsic and extrinsic parameters as input and provides road parameters i.e. road width, curvature of the road, vehicle's lateral displacement with respect to ego lane marking. This information along with the EKF predicted pose of the ego-vehicle $\bar{X}_{k+1}$ is used to project potential lane markings points on the map grid. These points are illustrated by green cells in center Fig.20 and represent source points. They are projected upto 10 m ahead of the vehicle with steps of 0.25 m. Central lane marking and road edges, signifying the side lane markings, are extracted from the grid map to perform lane-level map matching. They are illustrated by white cells in centre Fig.20 and represent target points for map matching algorithm. For each source point, corresponding lane marking target points (cell matches) on the map grid are selected using nearest neighbor search. 2D Iterative Closest Point (ICP) algorithm [33] is used to make pose correction between the sets of source and corresponding target lane marking points. EKF predicted pose $\bar{X}_{k+1}$ is taken as the initial guess for ICP. The corrected 2D pose that aligns source points to target lane marking points on grid map is introduced as a new observation for EKF.

ICP observation equation is defined in Eq. 3.

$$Y_{icp} = \begin{bmatrix} x_{icp} \\ y_{icp} \\ \theta_{icp} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + m_k \tag{3}$$

where x_{icp}, y_{icp} and θ_{icp} represent ICP corrected 2D pose of the vehicle in ENU coordinate frame and m_k represents the observation error. Assuming that m_k in the ICP observation is Gaussian distributed and the noise sources of position and orientation are independent, error covariance matrix Q_{icp} is given in Eq. (4).

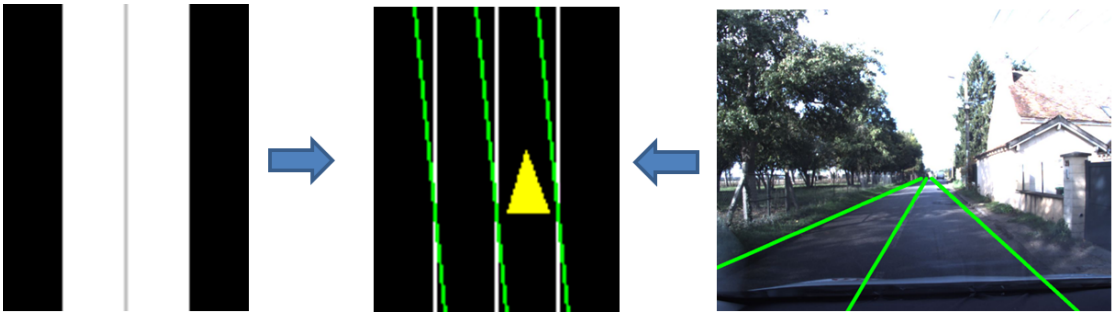


Figure 20: Lane level map matching on grid map with lane tracker. a) original map grid, b) lane marking information from image based lane tracker, c) extracted lane markings in white and lane tracker based lane markings projected in green with respect to ego vehicle's pose (yellow triangle).

$$Q_{icp} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & 0 \\ \sigma_{xy}^2 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{4}$$

While the observation is in the world ENU frame, it is more intuitive to define the longitudinal σ_{long} and lateral σ_{lat} standard deviations in the vehicle frame. The translational components of the Q_{icp} can then be computed as follows:

$$\begin{aligned} \sigma_x^2 &= \sigma_{lat}^2 \cos^2(\theta) + \sigma_{long}^2 \sin^2(\theta) \\ \sigma_y^2 &= \sigma_{lat}^2 \sin^2(\theta) + \sigma_{long}^2 \cos^2(\theta) \\ \sigma_{xy}^2 &= (\sigma_{lat}^2 - \sigma_{long}^2) \cos(\theta) \sin(\theta) \end{aligned}$$

σ_θ represents standard deviation of vehicle's heading. Detailed derivation of the covariance matrix can be referred to in Najjar et al. [34].

Fault detection scenarios are considered to verify if the ICP corrected pose is suitable for EKF observation. Error metric in ICP is defined by sum of squared distances between corresponding points. If the metric error does not fall below a threshold in a given number of iterations, the ICP is considered non convergent. For experiments, this threshold was set to $0.6\,\mathrm{m}^2$ for every respective pair of source and target points. In some cases, it is possible that the ICP is convergent but corrected pose is still incorrect, e.g. in case of convergence to local minima. To take this into account, a separate threshold is set for acceptable translational and heading correction, based on the assumption that the vehicle is headed parallel to the direction of road. The experimental values used for this translational and heading threshold were 10 m and 10° respectively. In case, the ICP is non convergent or the threshold is crossed, ICP observation for the EKF is skipped.

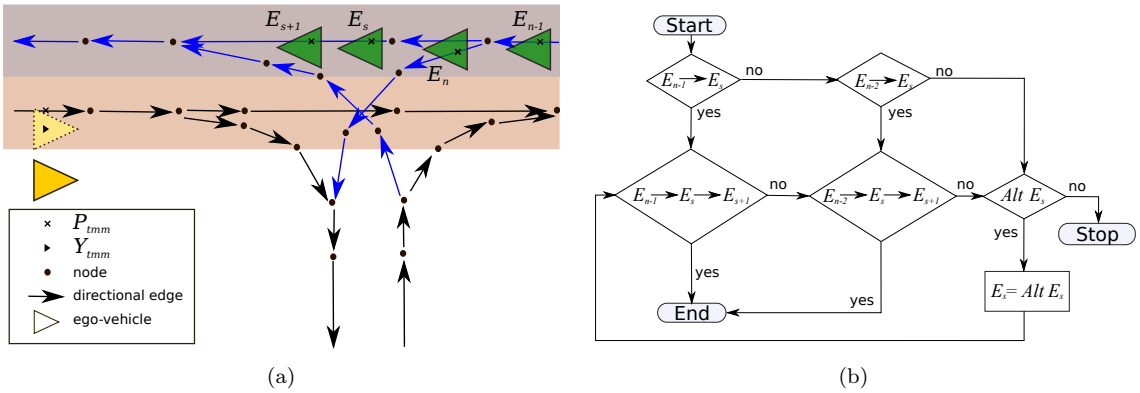
Topological map matching The topological map matching (TMM) algorithm uses if-then decision-rule approach [35] to snap the vehicle on to the most appropriate edge. Initially, the algorithm defines a search area of fixed radius around the estimated vehicle position. For experiments, this radius was set to 30 m. All the edges partially or completely part of this radius and oriented within 90° of the vehicle's heading are shortlisted. This way for any road, only the lanes feasible for the respected vehicle direction are considered. This is illustrated in Fig. 21a. For the ego-vehicle positions in green, only the edges in blue are considered for TMM.

With availability of new vehicle pose estimate, TMM algorithm snaps on to the closest edge to vehicle's position. Along the stretch of snapped edge E_n, the algorithm orthogonally projects the vehicle's position on the edge and selects it as the snapped point. The snapped point represents topologically map matched position P_{tmm} as represented by a cross in Fig.21a for respective ego-vehicle pose. For every new edge snapped E_s, the algorithm verifies that the current snapped edge E_n, has a feasible path to the new edge. If not, TMM algorithm enters a conflict resolution mode. This conflict may occur near an intersection or two separate roads running close by. In such a scenario, it is unknown whether E_n or E_s are incorrect (Fig. 21a illustrates a TMM conflict scenario where E_n is incorrectly identified). Thus, TMM algorithm awaits a subsequent edge E_{s+1} to correct the map matching and resolve the conflict. Subsequent edge is not necessarily the next edge in the map. In straight sections of road, an edge can be of significant length. If the vehicle moves some distance from the conflict position, the same edge can be subsequent edge. We set this distance to 4 m signifying the vehicle has moved at least its own length.

The conflict resolution is illustrated in the flow diagram in Fig. 21b. The algorithm checks if any of the previous two edges, E_{n-1} or E_{n-2}, has a feasible path to the new edge E_s and then to the subsequent edge E_{s+1}, this holds true for conflict shown in Fig. 21a. Otherwise, if that is not the case, alternate new edge, *Alt* E_s from shortlisted ones is considered. Candidate alternate edges are part of the search areas and they are considered one by one in order of increasing proximity.

Should no feasible path be found with alternate edges as well, the TMM conflict is not resolved. The TMM is stopped and re-initialized with an warning.

Since TMM algorithm operates by snapping on to an edge, it always provides information about vehicle's position (P_{tmm}) strictly on the center of the lane. However, the vehicle's actual position can be anywhere along the width of the lane. In consequence, making a periodic TMM observation as a Gaussian distribution with mean position value P_{tmm} will be inappropriate. Therefore, EKF observation is only made when the estimated pose of the vehicle is out of lane. To represent this, TMM observation to EKF is shown by a dashed line in Fig. 19. An example



(a) (b)

Figure 21: Topological map matching a) Topological map at a 3-way intersection. Nodes and edges are defined on the center of lane respectively. Green triangles illustrate position of the ego-vehicle at four instants representing a TMM conflict scenario; TMM relevant edges are shown in blue. Yellow triangle represents a different case where the ego-vehicle's estimated position is outside of lane, and the position shown by dotted triangle is the TMM observation, b) Flow diagram of conflict resolution in TMM.

scenario is illustrated in Fig. 21a where yellow ego-vehicle's estimated position is outside of lane. TMM observation Y_{tmm} is determined by computing point (x_{tmm}, y_{tmm}) on the lane section of map matched edge that is closest to estimated vehicle's position. This point is represented by the black triangle in pale yellow ego-vehicle. The TMM observation equation (Eq. 5) is defined in the ENU coordinate frame, similar to ICP observation Eq. 3. n_k represents the TMM observation error.

$$Y_{tmm} = \begin{bmatrix} x_{tmm} \\ y_{tmm} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + n_k \tag{5}$$

The covariance matrix Q_{tmm} is defined in the same manner as ICP covariance matrix Q_{icp} , except with only position components. The observation for EKF is skipped if the TMM algorithm is in a conflict mode.

The developed localization methodology and it's evaluation was conducted in Grenoble, near Inria's center, and the results of experimentation were published in IEEE ICRA2020 [36]. To compare different levels of sensor fusion, three use cases were studied: with both the map matching algorithms, only lane level map matching algorithm and using neither of them. It was shown that the two algorithms together can robustly localize the vehicle within the lane. Moreover, using visual tags at four points of interest, measurements were conducted and improved accuracy of the localization of the vehicle was established. The details of experimentation can be viewed in the publication.

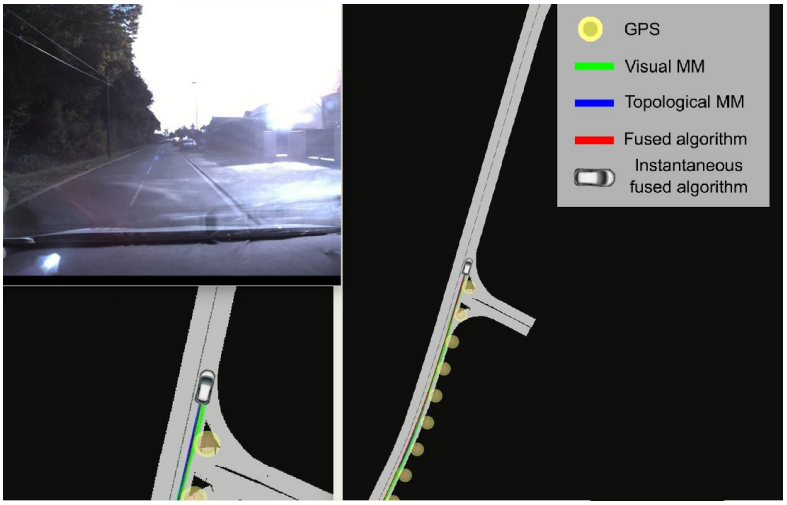


Figure 22: Online vehicle localization results on a segment of Route du Bray

The testing and validation of the presented localization approach was also carried out in Rambouillet. Figure 22 shows a snapshot of the localization results near one of the intersections. The two map matching algorithms together robustly localize the vehicle and assist in environmental perception and behavioral planning on Route du Bray.

3.2.3 Perception

Perceiving the surrounding environment and extracting adequate information is critical for scene understanding and decision-making. The autonomous vehicle must assess the traffic situation and respond to concerns such as: where is the lane and road with respect to vehicle? Is the road curved or straight? Are there any surrounding obstacles? If yes, are they static or dynamic? Is there a risk of collision to the obstacles?

We primarily tackle the perception problem using camera and LIDARs. In Autonomous vehicles, cameras are widely used for lane and road detection while the use of LIDARS, with its versatility to create a dynamic 3D map of environment, is common for object detection.

NVIDIA DriveWorks Lane detection module is used to detect and identify lane line markings. Based on camera image and its parameters, DriveWorks uses deep learning algorithms for lane markings detection, an example is illustrated in Fig.23a. While NVIDIA DriveWorks delivers great results during varying lighting conditions, it still has limitations. The module does not always recognize road boundaries (Fig.23b, 23c), a distant lane marking may be mistaken as an ego lane marking (Fig.23d), or thin water trail or road repair may be mistaken as a lane marking (Fig.23e).

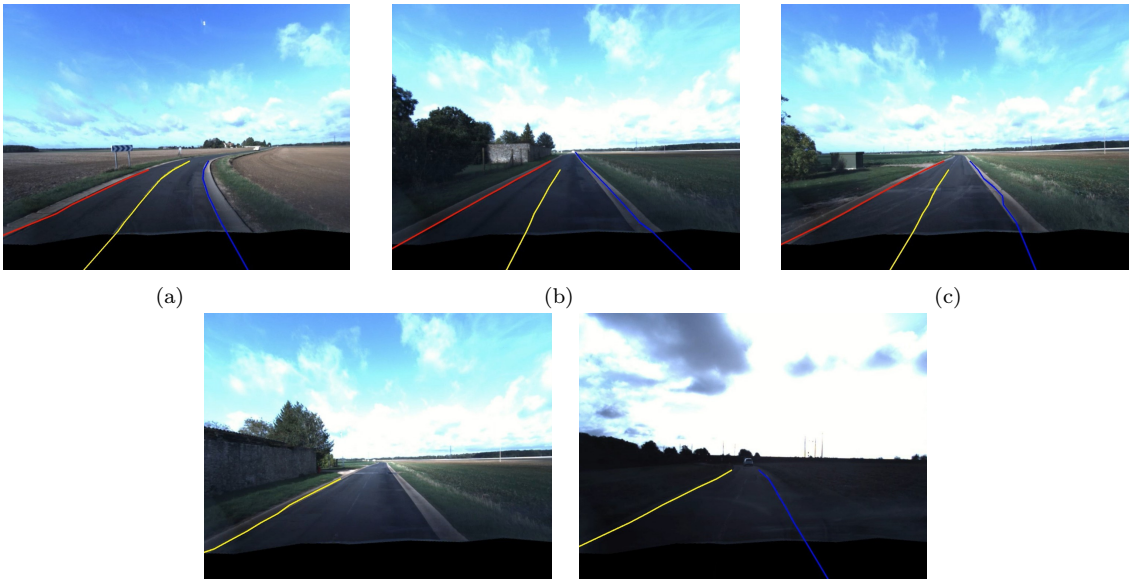


Figure 23: NVIDIA DriveWorks lane marking detection.

Curve estimation To address these limitations, we fuse information from lane detection, Velodyne (LIDAR), odometry and offline map for a reliable curve estimation, illustrated in Fig. 24.

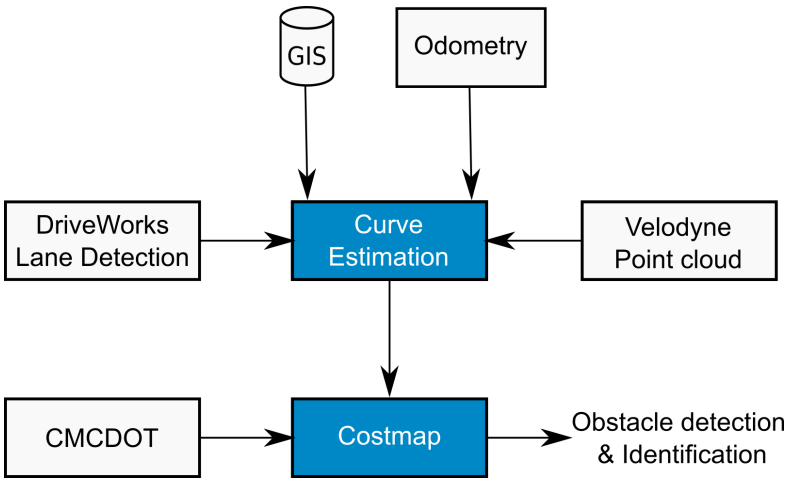
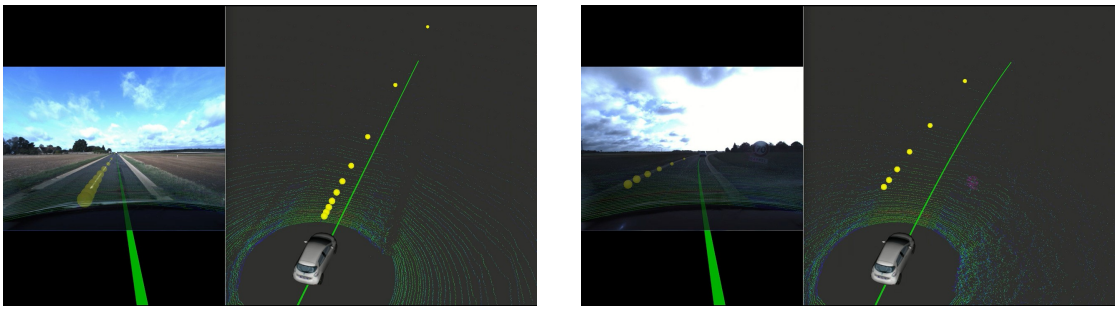


Figure 24: Overview of the curve estimation and object detection approach.

Initially, information from NVIDIAÂ® DriveWorks is translated in 3D with help of LIDAR. DriveWorks provides lane marking information as pixel positions on the image. To estimate the depth of respective pixels, Velodyne point cloud from the LIDAR is projected onto the camera image. Using pixel position and

LIDAR depth, position of lane marking points is estimated in 3D based on visual geometry. The yellow spheres in Fig.25 represent the estimated ego lane marking points in 3D. They are also projected on the camera image. Due to unreliable detection of lane markings at road boundaries (as can be seen in Figures 23b, 23c) of Route du Bray, curve estimation is performed based only on the left ego lane markings.



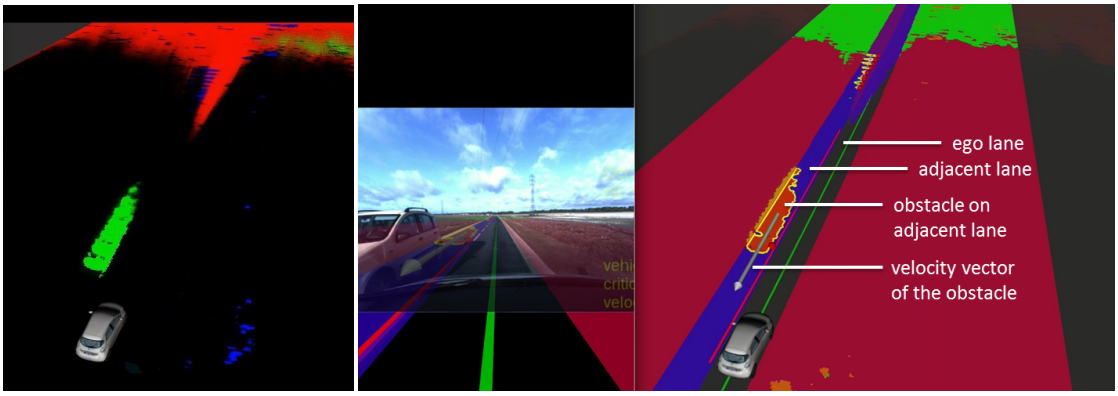
(a) (b)

Figure 25: Curve estimation based on NVIDIA DriveWorks and velodyne pointcloud.

Next, the ego lane makings are filtered before they are fused for road curve estimation. Based on initial or latest curve estimation, the lane marking points are filtered to ensure false ego lane detections such as the ones shown in Figures 23d, 23e, 25b are rejected. If any of the left ego lane makings are further than one lane width apart, the respective DriveWorks input is rejected. Moreover, these lane marking points are stored in odometry frame for preceding few seconds and within fixed distance to the vehicle's position (15m behind and 80m ahead of the vehicle). Thus, curve estimation does not only take into account the latest input from Driveworks but also the inputs collected over short period of time. These collective points are then used to estimate the curve using a second order polynomial defined in vehicle's base frame.

Finally, Extended Kalman Filter with the state vector comprising of co-efficients of the second order polynomial is implemented that fuses the lane markings based curve estimation with odometry and offline map. The prediction of the state is estimated with the odometry input. Similar to lane markings based curve estimate, a second order polynomial is implemented using offline map, localization in the map and the uncertainty in the longitudinal position of the vehicle. The offline map and lane markings based curve estimate are fused as the measurement for updated state estimate. In Fig. 25, green line illustrates the resulting estimation of curve on the ego lane center. The filtering and fusion of map and odometry helps overcome the specific shortcomings of the DriveWorks framework. Even with erratic or faulty ego lane marking detection, curve estimation remains reliable.

Obstacle detection The information from curve estimation is overlayed on the CMCDOT state grid to generate a costmap. The CMCDOT, in-house development of Inria-Chroma team, is a generic Bayesian Perception framework, that estimates a dense representation of dynamic environments and the associated risks of collision [37].



(a) (b)

Figure 26: Costmap generation a) CMCDOT state grid, b) illustrated costmap along with curve estimation.

Figure 26 shows an instant of CMCDOT state grid, illustrated cost map and camera image on Route du Bray. The CMCDOT state grid contains free, static, dynamic and unknown space represented in black, blue, green and red respectively. For cost map, we assign the lowest cost to free space and maximum cost to dynamic space. Curve estimation is translated into ego lane and adjacent lane with zero cost assigned to ego lane. For the costmap, the costs from cmcdot and state grid are summed. In Fig. 26b, the black, blue and maroon regions surrounding the vehicle represent free space on ego lane, adjacent lane and no-road region respectively. Cells with different colours on the lanes indicate static, dynamic obstacles or unknown occupancy, i.e., in case of distant or occluded regions. Contours are generated on the ego and adjacent lane obstacles, their centroids are computed, and their positions and critical distance the ego vehicle are determined. CMCDOT velocity grid helps estimate velocity of the respective contours.i.e. road obstacles.

Another instant of costmap is shown in Figure 27 near the entrance of the tunnel. In this case, two road obstacles are detected and illustrated. Ego-lane obstacle is represented in green and identified as vehicle ahead. It's velocity is in the same direction as that of ego vehicle and is coming to halt. The critical distance to the obstacle is determined to be 6.7m. The adjacent lane obstacle is identified as incoming traffic with help of it's velocity vector. Approaching vehicle is only recognized if the obstacle's velocity vector is directed towards ego vehicle. Thus,

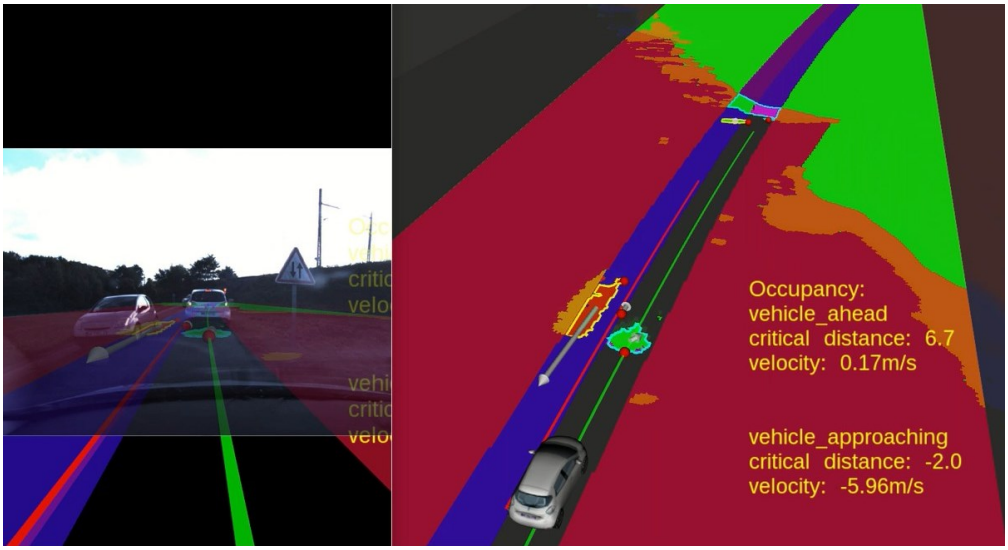


Figure 27: Object detection and classification illustrated on the cost map.

an overtaking vehicle is not considered an approaching one. In case of incoming traffic on Route du Bray, vehicles on both lanes cross each other while maintaining safe lateral gap between them. Therefore, for critical distance of an approaching vehicle, closest lateral distance to the ego vehicle is determined.

To conclude, the developed knowledge base for the navigation of the vehicle on Route du Bray is presented. Before moving on to the decision-making and control of the vehicle, the information available for reasoning is summarized.

1. metric and topological map of Route du Bray
2. map-relative localization of the vehicle
   (a) estimated vehicle's pose in the metric map
   (b) estimated vehicle's pose in terms of topological node and edge
3. curve estimation
   (a) road curvature estimation ahead of the vehicle
   (b) vehicle's displacement to center of the lane
4. obstacle detection on the road and classification
   (a) classification of obstacles on ego or adjacent lane
   (b) position and velocity of the obstacles
   (c) critical distance of the obstacles to ego vehicle

3.3 Hierarchical Decision-Making

Hierarchical decision-making is commonly split intro three tiers:

1. Route planning
2. Behavioral planning
3. Motion planning and control

3.3.1 Route planning

Route planner addresses high level mission objectives that deal with finding the path to the final destination. Given initial and target position, developed route planner uses offline topological map and Dijkstra's algorithm to define shortest route for the mission. The route is generated in terms of topological edges. This route planning informs in advance how long the route is and if the vehicle will cross the tunnel or any intersections or any addition offline information stored in the map. Fig. 28 shows part of topological route for crossing the tunnel in the direction of Gare de Gazeran superimposed on the map database. Topological route is illustrated in black, nodes in circles and directional edges in arrows.

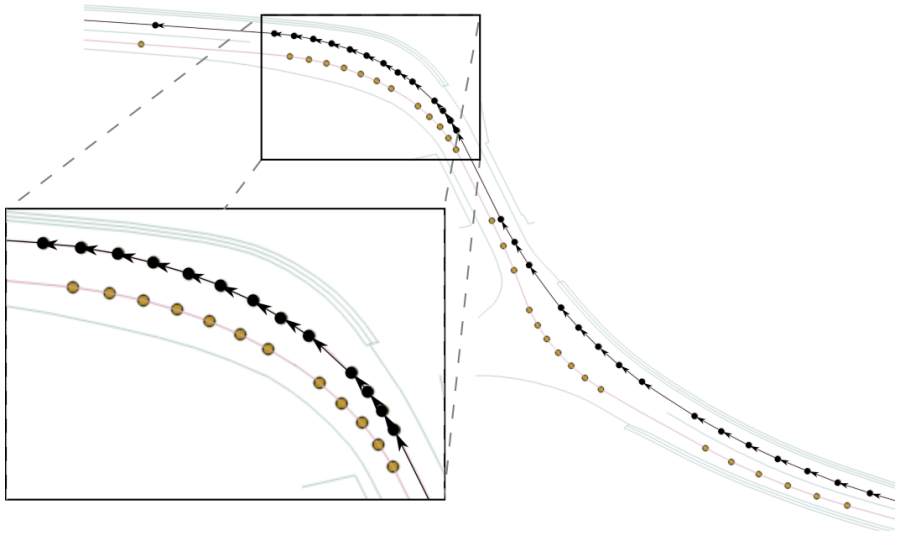


Figure 28: Route planning: topological route for crossing the tunnel in direction of Gare de Gazeran.

3.3.2 Behavioral planning

The behavioral planner is responsible for manoeuvre decision making and ensures that vehicle follows required road rules and interacts with other road obstacles (agents) accordingly. Manoeuvre selection is incorporated using Hierarchical Finite State Machines(FSM), illustrated in Fig. 29. While the autonomous navigation is active, FSM is either in *Drive* state, *Exception handling* in case of any exceptions or *Goal reached* when the destination has reached. With the aid of map-relative localization, the behavioural planner can accurately estimate vehicle's position in the next 10 s or next 100 m. Therefore, the planner can transition to the appropriate state such as *Approaching tunnel*.

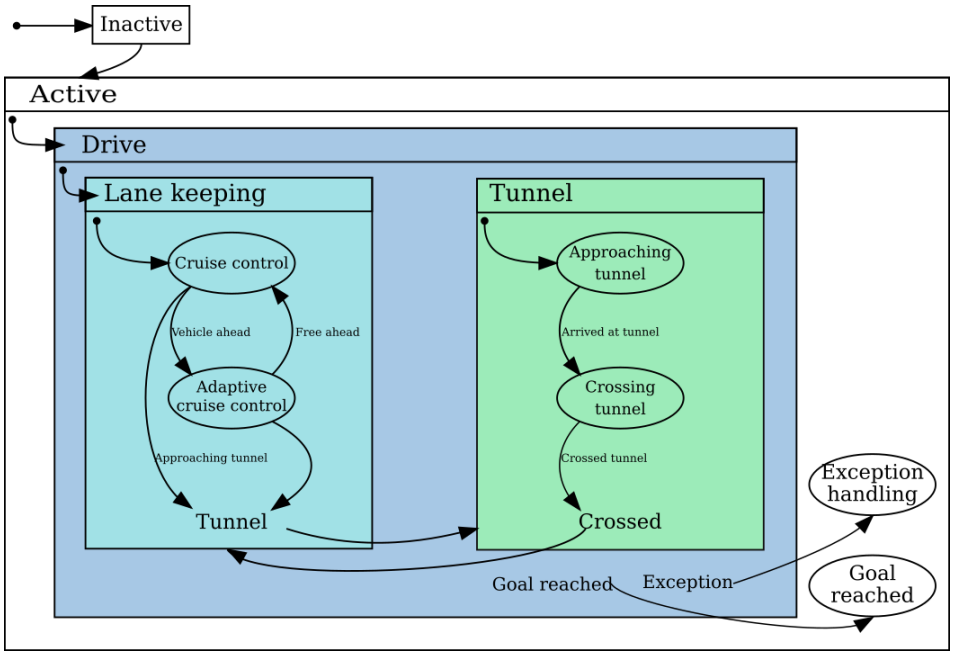


Figure 29: Hierarchical finite state machine for autonomous driving on Route du Bray.

For autonomous driving on Route du Bray, two main FSM states have been developed. Default state is set to *Lane keeping* in which obstacle information enables the vehicle to transition from *Cruise control* to *Adaptive cruise control* in case of road obstacles nearby. *Lane keeping* transitions to the *Tunnel* state when the vehicle nears tunnel. And once crossing the tunnel, FSM transitions back to *Lane keeping* state. All the developed states are discussed in detail in the next section, sec. 3.3.3.

3.3.3 Motion planning and control

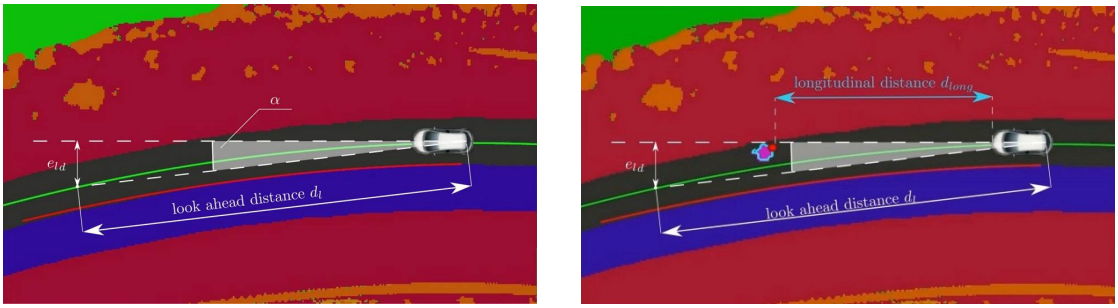
The motion planner generates appropriate paths and sets of actions to achieve local objectives specified by behavioral planner. This planner is responsible for executing manouevre in collision-free, efficient and comfortable manner for autonomous vehicle. Motion planning is closely connected to manoeuvre execution and drive control and are further explored together.

Two main modules of motion planning and control are the primary focus of development:

- Lane keeping - cruise control and adaptive cruise control
- Tunnel - approaching and crossing tunnel

Lane keeping The lane keeping module focuses on keeping the vehicle within the ego lane. Cruise control is the simplest case of lane keeping that is active when there are no surrounding obstacles. The local motion planner follows the path defined by the centre of the lane. The information concerning the road or lane geometry is provided by curve estimation module. When obstacles are present, lane keeping transitions to adaptive cruise control (ACC). In this case, local motion planner optimizes the path to ensure collision-free and safe path while still keeping the vehicle in it's lane.

Cruise control A common approach of pure pursuit based path tracking algorithm is implemented in cruise control. Fig. 30a illustrates geometry of pure pursuit algorithm on the costmap. The algorithm determines a point on the curve estimation at the look ahead distance d_l in front of Zoe and computes required steering action proportional to the lateral displacement represented by e_{l_d}. Pure pursuit enables lateral control of the vehicle and is primarily actuated by the steering wheel.



(a) (b)

Figure 30: Geometry of pure pursuit algorithm.

The developed algorithm has been tested in simulation as well as validated on Zoe autonomous vehicle. Autonomous vehicle testing has been conducted with respect to lane keeping capabilities in controlled environment at low speed of 20-30km/h. For autonomous control lane keeping tests have been conducted at the Transpolis facility. In comparison with Route du Bray, the curve estimation at Transpolis is carried out without the availability of an offline map, meaning the only curve estimation information, and hence lane geometry, is available via online onboard sensors, the camera and lidar. On the other hand, the test facility has an infrastructure in ideal conditions, where lanes are wide and the lane markings are well defined. The curves, however, at the Transpolis facility are sharper than all the curves on Route du Bray. Hence, lane keeping ability of the autonomous vehicle at curves is well tested with respect to the original road.

Figure 31: Transpolis autonomous vehicle testing facility.

The vehicle testing is conducted on the perepherique route marked in green in Fig. 31, the complete loop is about 2km long. Autonomous control is tested and validated on the complete route without interruption and vehicle reliably kept its lane in both the directions. While there are clear lane markings available on both sides, to keep it similar to Route du bray, we only use left ego lane marking for curve estimation. To simulate the scenario for Route du Bray for a 2m wide lane, test runs are also conducted with the vehicle close to the left lane marking.

Figure 32 shows a screenshot of one of the cruise control test runs on Transpolis test track. On the right, one can see third person view of the autonomous vehicle with real time illustration of curve estimation, cost map and control commands on the left. The details and video recording of the tests can be viewed online[1].

Adaptive cruise control In case of surrounding obstacles, the lane keeping transitions to adaptive cruise control (ACC). ACC continues to keep the original pure pursuit to follow the center of the lane. Additional to the original pure pur-

[1]https://www.youtube.com/watch?v=7nRGsqk2j5A

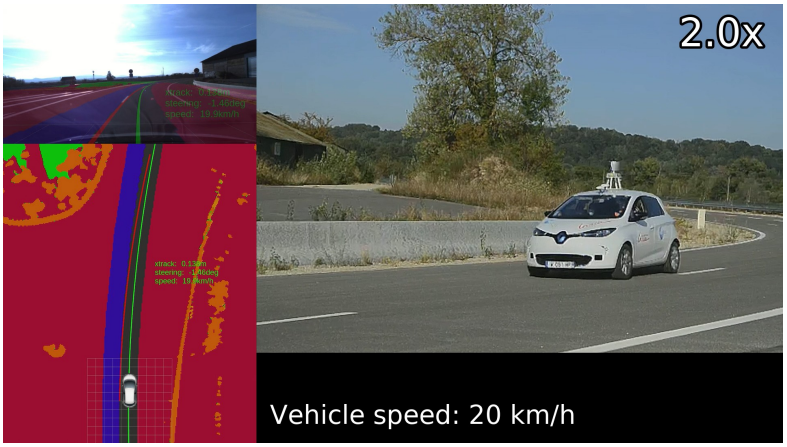


Figure 32: Cruise Control testing at Transplis.

suit control, ACC implements longitudinal control to maintain a safe distance to obstacles. Fig. 30b illustrates the longitudinal distance, d_{long} to ego lane obstacles.

The distance of Zoe(ego vehicle) to the obstacle is provided by the perception module. For a reliable implementation, the presence of obstacles is verified for multiple iterations before the ACC is triggered. Once ACC is active, the speed command for Zoe is gradually slowed to zero at at a steady deceleration rate. The static obstacle scenario on ego lane was tested at Transpolis testing facility. Figure 33 shows an snapshot of recordings of ACC testing with the static obstacle on ego lane, the vehicle gradually comes to halt before the inflated balloon. In this shown instant, the Zoe's speed has reduced from 20km/h to 3.4km/h and the longitudinal distance to the balloon is recorded to be 7.6m. The videos of the test runs can be viewed online[1],[2].

The adaptive cruise control with dynamic obstacles is developed and tested in simulation. In case of dynamic obstacle on ego lane, the vehicle must reduce its velocity to match that of preceding vehicle as well as maintain safe distance. The control action depends on two factors, the longitudinal distance between Zoe and ego lane obstacle and the difference between the velocities of Zoe and the obstacle, denoted as Δv. The longitudinal acceleration $a(t)$ of the vehicle is then determined by

$$a(t) = K_p(d_{long} - d_{safe}) + K_v \Delta v$$

where K_p is the gain to reduce the distance between Zoe and obstacle to a safe distance d_{safe} and K_v is the gain to match the velocity of the preceding vehicle. Figure 34 shows an instant of ACC simulation where Zoe maintains a significant

[2]https://www.youtube.com/watch?v=Qovb1K1agmw

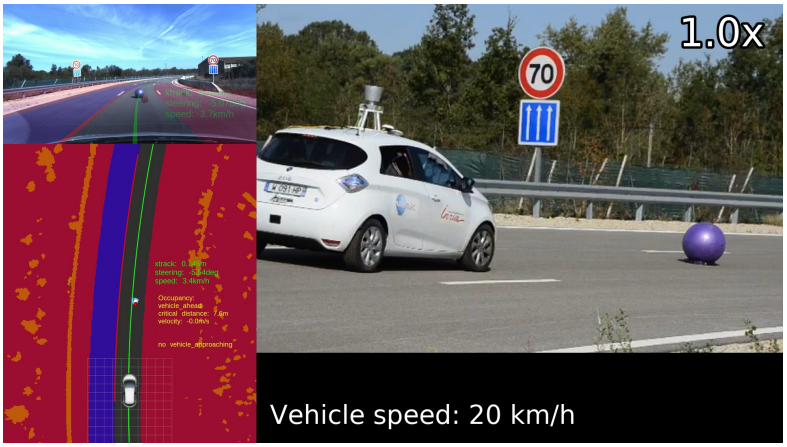


Figure 33: Adaptive cruise control testing at Transplis.

distance of about 20m to the ego lane obstacle. The simulation recordings can be viewed here[2].

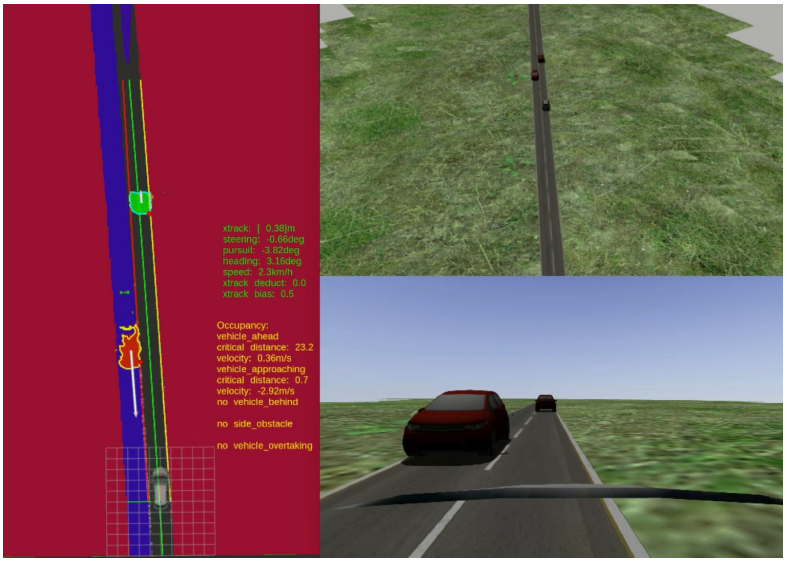


Figure 34: Adaptive cruise control with dynamic obstacles in Gazebo simulation.

In case of dynamic obstacle on the adjacent lane, the trajectory of the ego vehicle is altered to maintain a safe lateral distance to oncoming traffic. This response is very specific to Route du Bray. Since the lanes are very narrow, simply being within the lane is not sufficient for obstacle collision. Both the ego vehicle and oncoming traffic must displace to their respective right to ensure safe crossing. Similar to ego lane obstacle, for a reliable implementation, the presence of adjacent lane obstacle is verified for multiple iterations before the ACC is triggered. Once

ACC is active, the ego vehicle laterally displaces itself.

This lateral displacement is defined by additional trajectory that maintains lateral offset to the original path. It is realized with path defined by parametric curve. For both x and y displacement from the center of the lane, 5th order splines sets are used. The parameters of the equations vary from zero to one.

$$\begin{aligned} x(u) &= a_0 + a_1 u + a_2 u^2 + a_3 u^3 + a_4 u^4 + a_5 u^5 \\ y(u) &= b_0 + b_1 u + b_2 u^2 + b_3 u^3 + b_4 u^4 + b_5 u^5 \\ & u \in [0, 1] \end{aligned}$$

where u(0) refers to the time when ACC is triggered and u(1) corresponds to the time when the ego vehicle path has laterally displaced to maximum value. The boundary conditions are defined as

$$\begin{aligned} x(0) &= 0 & y(0) &= 0 \quad y(1) = w \\ \dot{x}(0) &= \dot{x}(1) = v & \dot{y}(0) &= \dot{y}(1) = 0 \\ \ddot{x}(0) &= \ddot{x}(1) = 0 & \dot{y}(0) &= \ddot{y}(1) = 0 \end{aligned}$$

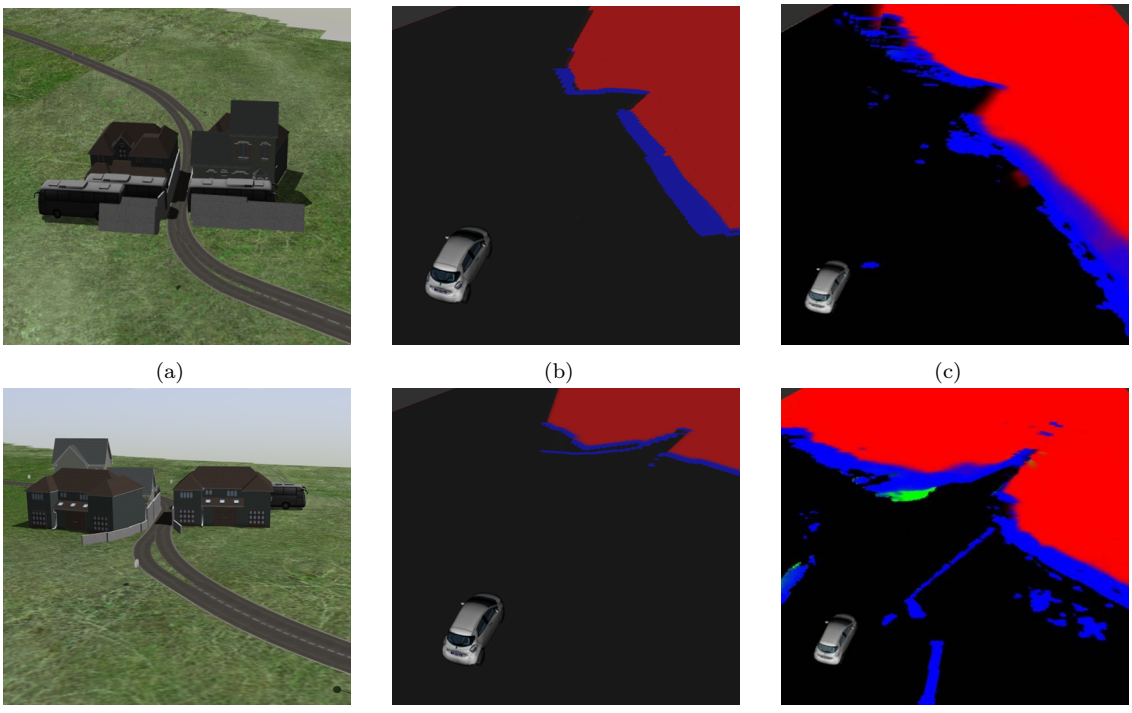
where w denotes the lateral width displacement and is set to -0.5m and v_0 denotes the linear speed of the vehicle. An example of initiated lateral displacement is shown in Fig.34. Once the approaching vehicle(s) has crossed, the ego vehicle returns back to center of lane with trajectory defined by similar parametric curves.

Tunnel The manoeuvre execution at the tunnel is mainly divided into two states:

1. Approaching tunnel
2. Crossing tunnel

In *Approaching tunnel* state, while maintaining lane keeping the vehicle gradually comes to halt before it arrives at tunnel. *Crossing tunnel* state consists of three main sequential steps:

- generate path through the tunnel using CMCDOT state grid, offline map and vehicle's pose.
- await and confirm the signal status via V2X unit.
- run dynamic window approach (DWA) planner to follow the path



(a) (b) (c)

(d) (e) (f)

Figure 35: Simulation for tunnel crossing. Gazebo simulation environment (left), CMCDOT state grid in simulation (center) and in real conditions (right). Direction of travel coming from Gare de Gazeran(a),(b),(c) and going towards the Gare de Gazeran(d),(e),(f).

Simulation environment for the tunnel is created in Gazebo using the offline map, see sec. 3.2.1. The static obstacles are additionally planted to mimic the original tunnel environment. Figure 35 shows gazebo environment, CMCDOT state grid generated in simulation and in real condition. Fig 35a presents the tunnel setup for direction of travel coming from Gare, with Fig. 35b and 35c showing the CMCDOT state grid when the vehicle arrives at the tunnel in simulation and in real environment respectively. The static obstacles are placed such that the similar features are available that can be used to identify the entrance of the tunnel. Similarly, tunnel environment is set for the direction of travel going towards the Gare, in Fig. 35d, 35e, 35f.

To generate path, entrance to the tunnel is identified on the CMCDOT state grid. For this static obstacles, on the left and right of the tunnel are classified and the closest point between the two sides is identified as entrance. The coordinates for entrance and exit to the tunnel are pre-identified in available map. The identified entrance points are then associated with the stored coordinates with additional information from map-relative localization of vehicle. Path to cross the tunnel is generated with respect to vehicle's current pose. Figure 36 shows the path in green on state grid for both directions. At this instant the other end (exit) of the tunnel is not identifiable on the state grid. Still, the generated path is a good estimate to guide the vehicle through the tunnel and keep the vehicle on road. When the vehicle is inside the tunnel and the exit is visible (on the state grid), the path is updated to adjust its alignment, see Fig. 37.

(a) (b)

Figure 36: Path generation using the CMCDOT state grid when the vehicle arrives at the tunnel. Direction of travel coming from Gare(a), going towards the Gare(b).

Once the path is generated, and confirmation is received that the signal at the tunnel, dynamic window approach (DWA) path planner is launched. The DWA planner is Inria's in-house developed software, developed by Thomas Genevois in team Chroma. DWA utilizes the CMCDOT occupancy grid to compute commands

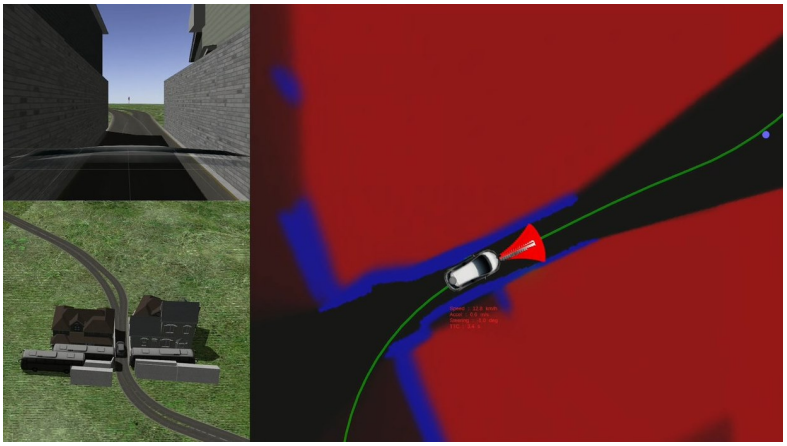


Figure 37: Tunnel path corrected and aligned when the vehicle is inside the tunnel.

for collision-free trajectories following the target path. Figure 38 shows an instant of the DWA planner in simulation. The red cells ahead of the vehicle represent trajectories with potential collisions, while white and grey cells represent collision-free trajectories of varying costs. The planner selects the commands associated with lowest cost and sends the throttle and steering command to low level controller.

Video recordings of the tunnel crossing in simulation can be viewed online[3].

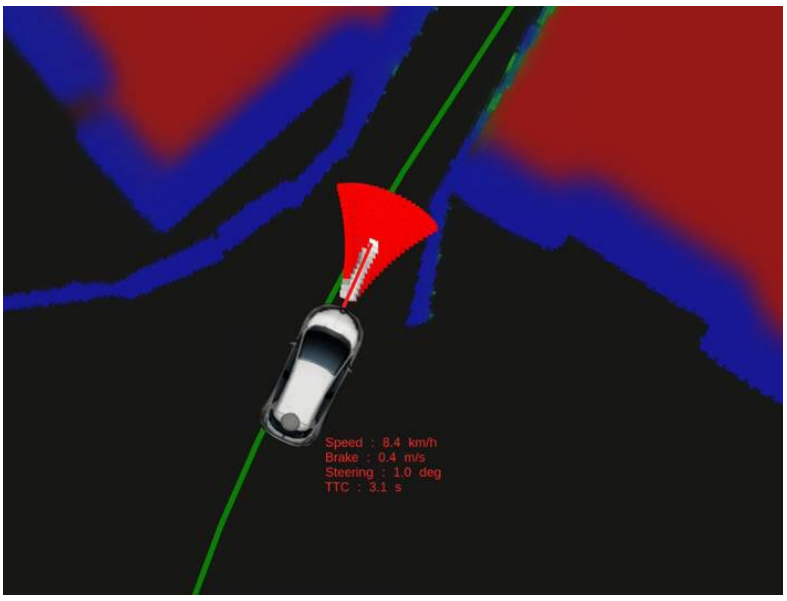


Figure 38: Dynamic windo approach planner.

[3]https://youtu.be/ovzY7sj3ksA

3.4 V2X Communication

For V2X communication, Neavia's on-board unit (OBU) is installed in the Zoe. Two main tasks are carried out: mission exchange and traffic signal interface.

3.4.1 Mission exchange

Zoe communicates with the fleet management via the opaque messages with the V2x server to accept mission for the autonomous navigation. The vehicle accepts mission commands from the the fleet management and in return sends mission status. Figure 39 gives an overview of the mission exchange with the OBU.

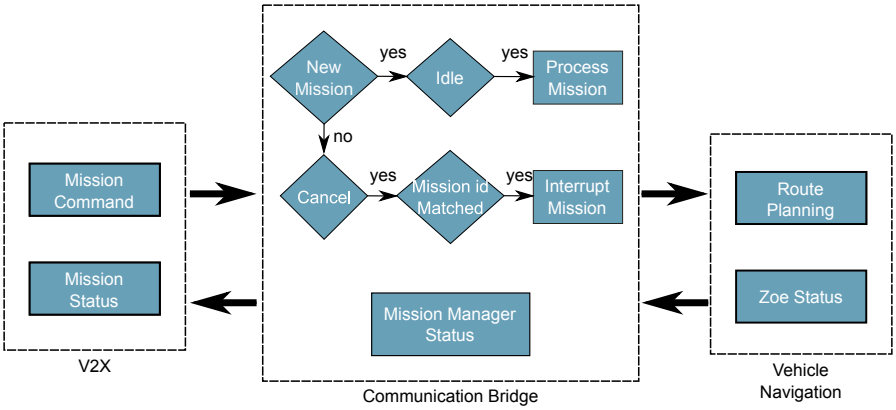


Figure 39: V2X communication for mission exchange.

The fleet management either sends a new mission or sends the cancel request via mission command. Zoe only accepts a new mission if it is idle, .i.e. not already running any mission at the moment and provides the relevant information to route planning module. In case, the fleet management sends a cancel command, Zoe confirms that mission id of the current mission and cancel command matches and then interrupts the mission by providing relevant information to route planning.

In return Zoe sends the status of the mission. This includes current mode whether it is manual or autonomous and manager state, whether the mission is disabled, in process or completed. The mission status also includes vehicle speed, orientation and the route. The position is by default sent by the OBU, provided by the GPS.

3.4.2 Traffic signal

Zoe must read the traffic signal when it arrives at the tunnel and only cross when the signal is green. For this, Zoe reads traffic signal status via OBU, see Fig. 40. Standard SPAT message (Signal Phase and Timing) is read and processed. If the

signal to the entrance of the tunnel is green, and the time left to cross the tunnel is sufficient before the signal changes, the 'go' command is communicated to the Tunnel crossing module.

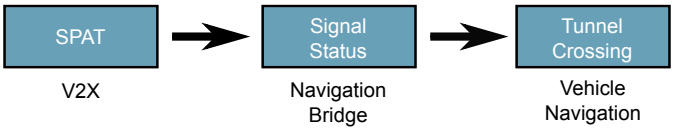


Figure 40: V2X communication for traffic signal interface.

3.5 Conclusion

The developed system architecture for the autonomous vehicle navigation in Project TORNADO has been presented in detail. The problem of vehicle navigation has been strictly developed for Route du Bray in Rambouillet.

Various segments of autonomous driving have been built including hierarchical decision-making, vehicle localization, perception of the environment, V2X communication, path planning and control of the vehicle.

The various challenges involved in scene understanding, situational assessment that are specific to Route du Bray have been addressed and solutions have been presented. Autonomous vehicle testing has been conducted with respect to the lane keeping capabilities in controlled environment at low speed. All the developed algorithms have been validated on Zoe autonomous vehicle (except the tunnel crossing, which has been simulated in Gazebo) and are available to further improve upon and build high speed autonomous capabilities.

4 Navigation Methods to Cross Roundabouts Safely - Heudiasyc UTC-CNRS

4.1 Introduction

Despite the tremendous growth of research regarding fully autonomous driving vehicles (AD) in the past few years, many safety critical scenarios, such as the crossing of roundabouts, are still open issues. Vehicle-to-vehicle and vehicle-to-infrastructure communications offer an appealing solution to handle such situations in a cooperative way.

In this document, we propose to adapt the concept of virtual platooning to the roundabout crossing use-case. This idea allows a single navigation strategy to handle complex scenarios such as intersection and roundabout crossings in a general and scalable fashion. First, we propose to generalize this approach not only to communicating AD vehicles, but also to manually driven (MD) communicating vehicles or even non-communicating ones.

In the second part of the work, we propose an extension of the aforementioned navigation strategy to a navigation scenario with one self-driving car among a flow of MD non-communicating and non-cooperative vehicles. One of the major challenges of this driving scenario is the safe navigation of AD on roads open to public traffic. Indeed, behaviors and intentions of MD vehicles are hard to predict and understand.

Our approach relies on High-Definition (HD) maps with lane level description which allows to predict the future situation thanks to the concept of virtual vehicles. This method handles safely collision avoidance and guarantees that no priority constraint is violated during the insertion maneuver without being overly cautious. The performance is evaluated with the SUMO simulation framework. A highly interactive vehicles flow has been generated using real data from the INTERACTION dataset. We also propose strategies to extend our algorithm to multi-lane roundabouts and report how these extensions behave in terms of safety and traffic flow. Finally, we show the feasibility of our approach performing real tests carried out with an experimental AD vehicle on a test circuit in real time. Our results show that this approach is easy to integrate into an embedded system and that it allows roundabouts to be crossed with a high level of safety.

This document is organized as follows: Section 4.3 presents the algorithm in the most generic case where all the aforementioned simplifications have been progressively removed. Finally, section 4.5 illustrates the main experimental results and in section 4.6 we discuss the main conclusions and future perspectives about the presented work.

4.2 Curvilinear signed inter-distance

The first step of the curvilinear representation is to get the lane followed by the vehicles. For an AD vehicle, it is simply given by the path planning module in the form of a list P=$(L_1, L_2, ...)$ of links to follow. In contrast, the driving lanes of the MD vehicles need to be estimated by a map-matching procedure. It is well known that map-matching ambiguities may arise when a road splits or when the vehicle changes lanes. We do the map-matching process using the method described in [1] where multiple candidate lanes can be simultaneously occupied by creating virtual instances of the MD vehicle, one for each lane possibly occupied. This technique will be detailed in section 4.3.

According to the notation introduced [38] and [1], the curvilinear abscissa s_{L_k} of a vehicle lying on the link L_k and map-matched to its i-th segment $l_k^{(i)}$ is computed w.r.t. the starting node $p_k^{(0)}$ of L_k as follows:

$$s_{L_k} = \sum_{j=0}^{i-1} \ell_k^{(j)} + \lambda \ell_k^{(i)}, \tag{6}$$

where $\lambda \in [0, 1]$ is a parameter to model the position along the map-matched segment $l_k^{(i)}$. In the rest of the chapter, s_{L_k} will simply be noted s when there is no ambiguity. Practical computation of the curvilinear coordinates can be found in [39].

As in the case of cooperative roundabout crossing, this representation is particularly useful to do virtual platooning for safely crossing an intersection with a vehicle driving on another road [40]. Given the path of the vehicle $P = (L_k, L_{k+1}, \ldots)$, the curvilinear abscissa to any node from the path, for example, the ending node of the link L_n, is computed as

$$s_n = \sum_{j=k}^{n} \mathcal{L}_j - s_{L_k}. \tag{7}$$

Equation (6) computes the curvilinear abscissa of a vehicle measured w.r.t. the start node of the current link L_k, while in equation (7) the quantity s_n is measured from the ending node n of link L_n backward to s_{L_k}. Note that s_n is negative and increases as s_{L_k} approaches the node n.

Let us consider two vehicles V_i and V_j with path P_i and P_j, respectively. If the two paths intersect, the notion of curvilinear abscissa can be used to compute the relative virtual gap between the two vehicles. Suppose that P_i and P_j intersect at a given node n, and let $s_{i|n}$ and $s_{j|n}$ be the curvilinear abscissa of V_i and V_j w.r.t. this node n as defined in equation (7). We define the virtual curvilinear signed

inter-distance between V_i and V_j as follows:

$$d_{i,j|n} = -d_{j,i|n} = s_{j|n} - s_{i|n}. \tag{8}$$

Note that $d_{i,j|n}$ is a signed inter-distance where $d_{i,j|n} > 0$ means that V_i is closer to n than V_j, *i.e.*, V_i is virtually ahead of V_j. Moreover, Figure 42 illustrates the computation of the curvilinear abscissa and distances.

4.3 Single-lane Roundabout Crossing with priority and interval occupancy

Problem Description and Contributions In [41], we have shown that using virtual vehicles along the lanes of an HD map is very efficient to predict the dynamic situation in a roundabout and to control the longitudinal behavior of an AD vehicle.

Then, an extension of this method to handle cases with both AD and MD vehicles is presented. However, some constraining hypothesis have been made about the MD vehicles behavior. In particular, we assumed that the localization of every MD vehicle is known with low uncertainty and that MD vehicles drive with a straightforward behaviors i.e. without breaking traffic rules, which is not always the case.

In addition, we considered also that direct communication exists between both AD and MD vehicles. This means that they are able to exchange information about their states respectively. Unfortunately, nowadays this assumption is unrealistic, because there are only a few MD vehicles on public traffic equipped with this kind of systems.

As a consequence of that, during autonomous navigation, AD vehicles have to estimate by their own on-board system the state of surrounding MD vehicles.

Finally, in the previously described approach we did not take explicitly into account traffic rules in roundabout navigation. In particular, the priority constraint between vehicles on the roundabout ring and incoming vehicles on entering branches has been disregarded.

In this section, we propose an extension of the previously discussed strategy to incorporate gradually the aforementioned conditions in order to provide a more general navigation algorithm to be exploited in real MD traffic driving scenarios.

Then, we provide an extended experimental evaluation to show the feasibility of our approach on a real-traffic scenario and to focus on the resolution of some issues that arise when virtual vehicle techniques are applied to roundabouts. The main questions of this part are listed hereafter:

- A safe, priority-preserving and not overly cautious decision method for one-lane roundabout crossing for an AD vehicle among an MD vehicles flow;

- An extension of the curvilinear coordinates virtual platooning to an interval-based curvilinear formalism to include both vehicles sizes and uncertainties in vehicles positions in the navigation algorithm;
- The use of the virtual vehicle concept to handle prediction of unknown intention of vehicles and to handle the application of virtual vehicle methods to roundabout cycles;
- A practical evaluation of the aforementioned algorithm in terms of safety and fluidity of the insertion with realistic simulations and real experiments;

The rest of this part is organized as follows. In the next section, we first introduce curvilinear formalism along with HD maps. Then, in section 4.3, we explain in detail the concept of virtual instances and its application to roundabouts use cases. In section 4.3, we discuss the main principles and rules that we use in our approach, while in section 4.3, we show how we take into account the priority constraints and the right of way. In section we explain the insertion maneuver for roundabout insertion maneuver in the case with MD vehicles involved. Finally, in section 4.4 we present the strategies to handle multi-lane roundabouts, followed by real experimental results in section 4.5.

Interval-based curvilinear inter-distances In this section, we present the intervals-based formalism to compute curvilinear inter-distances between several vehicles. Such concept can be seen as a generalization of point-based inter-distances that we used in [41]. The main idea of this step is to represent objects positioning w.r.t. the HD map with an interval rather than a point on the polyline. Such formalism is useful to better encompass uncertainty about positioning of MD vehicles. In particular, we exploit what was previously said in [41] and [38] about our HD map formalism and curvilinear coordinates computation to develop our new formalism.

In the real world, a single curvilinear point is not sufficient to represent the physical occupancy of a vehicle. Instead of using curvilinear coordinates, we propose to replace them with curvilinear intervals to better represent objects occupancy.

Based on the curvilinear formalism introduced in section 4.2, let us first generalize the virtual curvilinear inter-distance of equation (8) to the case of interval curvilinear abscissa. For an interval $[\underline{s}_i, \overline{s}_i]$, we define $\left[\underline{s}_{i|n}, \overline{s}_{i|n}\right]$ the lower and upper curvilinear abscissa of the vehicle w.r.t. a given node n using equation (7). As said previously, these quantities are negative and increase when the vehicle approaches the node n. We define the interval-based virtual curvilinear signed inter-distance $d^*_{i,j|n}$ between V_i and V_j, whose paths intersect at a node n, as the

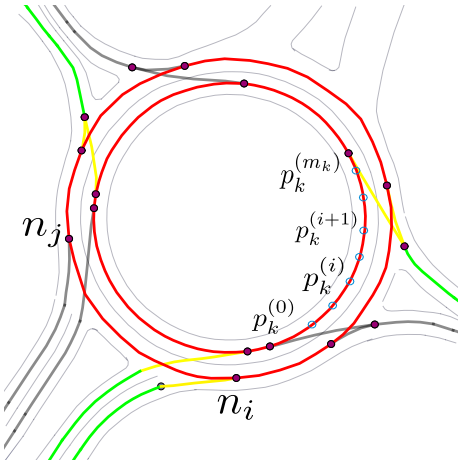


Figure 41: A roundabout with its HD map representation. The decision zones are green, the transition zones are yellow and the ring zone is red. The roundabout exits are blue. For a given link, the nodes and shape points are shown.

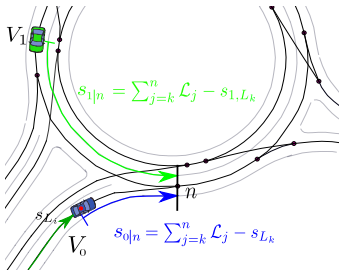
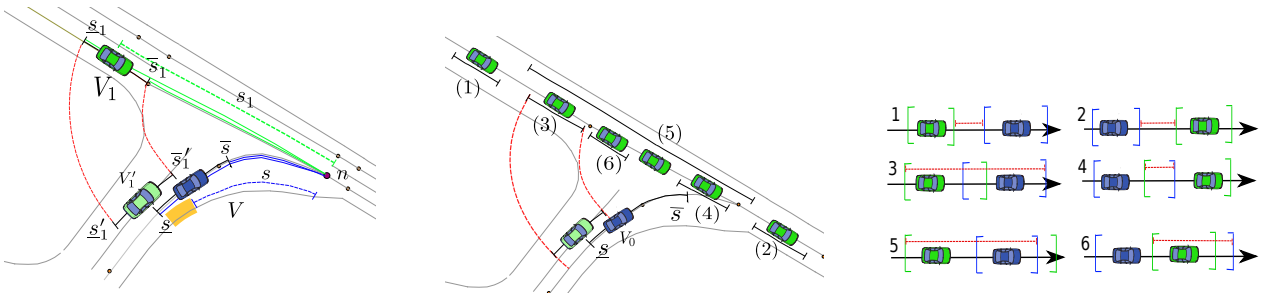


Figure 42: The curvilinear abscissa w.r.t. a given link and the relative distance computation to the intersection point as explained in section 4.3. The blue arrow pointing towards the intersection point indicates the sign of the distance.



(a) (b) (c)

Figure 43: (43a) Classical virtual platooning (dashed) and its extension to intervals (solid). V is the ego-vehicle and $V_1^{'}$ is a virtual instance of V_1. (43b) Situation with 6 other vehicles (all the possible relative locations) Occupancy is projected onto the road map. (43c) Intervals overlapping corresponding to Fig. (43b).

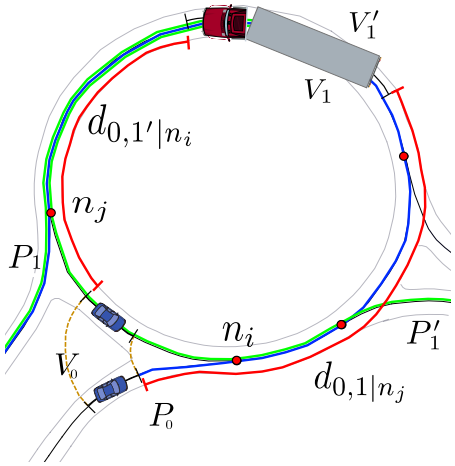


Figure 44: The truck trajectories are estimated considering several virtual instances of the same vehicle according to [1] assigning at each instance a possible trajectory. In this case, the truck can be both ahead and behind the AD vehicle as explained in section 4.3.

signed distance between the front of V_j and the back of V_i:

$$d^*_{i,j|n} = \overline{s}_{j|n} - \underline{s}_{i|n}. \tag{9}$$

Therefore, $d^*_{i,j|n} < 0$ means that w.r.t. node n, V_i is virtually ahead of V_j, i.e., they are in the first configuration illustrated in Figure 43c. It is important to note that $d^*_{i,j|n} > 0$ does not mean that V_j is ahead of V_i, indeed, any of the other configurations in Figure 43c would lead to this case. Contrary to the case where a single curvilinear abscissa is used, we have $d^*_{i,j|n} \neq -d^*_{j,i|n}$, i.e., this equation is not symmetric. To better understand this concept, Figures 43a, 43b and 43c explain such concept in the case of a simple T intersection. In the rest of the work, for simplicity, $d^*_{i,j|n}$ will be denoted $d^*_{i,j}$ when there is no ambiguity.

Unknown MD vehicles intentions in graph cycles In order to encompass all the possible behaviors of an MD vehicle, we consider that a single MD vehicle can be represented by several different virtual instances.

In this section, we exploit the concepts that have been already introduced in [41] to provide a method that can help to solve some situations of relative positioning inside the roundabout ring that arise in cyclic graphs. If we look carefully at Figure 44 and we apply the virtual platooning algorithm as in [1], we can see that the paths of V_0 and the instance V_1 of the truck have n_i as the first intersecting node (from the point of view of the AD vehicle V_0). This means that V_1 is virtually behind V_0 even if it is clearly not the case because the truck can be seen both behind and ahead V_0, depending on the point of view. Moreover, if one considers

long objects, as for example a truck with a trail, this ambiguity is more flagrant. This particular behavior is due to the circular shape of the roundabout ring and in some cases it may produce an erroneous representation of the scenario. In fact, V_0 needs to consider the presence of the truck during the insertion maneuver as a vehicle to follow, and it also needs to consider its presence as a possible incoming vehicle on the left side of the roundabout. To overcome this issue, if we consider again the instance $V_1^{'}$ that represents the same vehicle but with a different path, one can see that it has n_j as the first intersecting node with the path of V_0. This means that, in this case, V_0 is behind the instance $V_1^{'}$. This representation is the dual of the previous one, where we claimed that the instance V_1 of the truck was virtually behind V_0. As a consequence, in a roundabout we can have several instances of the same object with a different relative positioning w.r.t. the AD vehicle. In other words, an MD vehicle can be both virtually ahead and behind the AD vehicle. This method enables the AD vehicle to overcome the problem of the roundabout loop and allows it to consider all the possible MD vehicles configurations in the scenario.

Insertion strategy Let us assume that the state of an MD vehicle V_i is represented as

$$V_i = [\underline{s}_i, \overline{s}_i, v_i, P_i]\,, \tag{10}$$

where $[\underline{s}_i, \overline{s}_i]$ are the lower and upper bounds over the curvilinear occupancy of V_i encompassing both the size and the uncertainty bounds over its position estimate, v_i is its estimated longitudinal speed and P_i its predicted path.

The values of $\underline{s}_i$ and $\overline{s}_i$ are computed as explained in section 4.3. From the perspective of the ego AD vehicle, the state of a nearby vehicle is typically provided by a perception system able to detect, track and map-match.

Moreover, in this section we decide also to add the MD velocities v_i because it allows to better take into account the dynamic behavior of the MD vehicles. As we previously said in section 4.3, we considered an overly-simplified behavior model for MD vehicles. Adding information about MD vehicles speeds allows to better encompass both MD vehicles behavior and to implement a more complex roundabout insertion strategy that handles also priority constraints between vehicles. This will be detailed in section 4.3.

In this work, in order to be compatible as much as possible with most of the state-of-the-art road users detection algorithms, no other assumption has been made on the information about surrounding vehicles. In particular, the reader may refer to [42], where it is explained how the quantities in equation (10) are provided to our system.

We constrain the AD vehicle to navigate only on the outermost lane of the roundabout. In other words, we do not allow the AD vehicle to overtake and change

lane during the roundabout crossing. With this simplification, the navigation algorithm only needs to control the longitudinal motion of the AD vehicle to perform the task, the lateral control being done by path following.

Moreover, to cross a roundabout successfully, we need to take into account not only the safety inter-distance w.r.t. the vehicle ahead, but also the priority relationships in the roundabout scenario. In a roundabout, the priority lanes are situated inside the roundabout ring while the non-priority ones are in the entering branches. Bearing in mind such concepts, we infer three rules that would describe the ideal behavior that an AD vehicle should have. An AD vehicle:

R1) has to keep a safe inter-distance w.r.t. the vehicle ahead;

R2) has to respect the traffic rules (vehicles inside the roundabout have the right of way);

R3) should avoid stopping on the carriageway as much as possible.

This means that a vehicle on a non-priority lane is allowed to enter into the roundabout only if the insertion maneuver does not influence the behavior of another vehicle with a higher priority rank. In other words, the entering vehicle is not allowed to force a priority vehicle to decrease its speed. A priority vehicle follows its reference speed profile and performs an inter-distance regulation only with respect to vehicles that have the same priority level. It is also desired that the AD vehicle makes an insertion as smooth as possible without making a stop at the entrance of the roundabout which requires it to anticipate the behavior of the other vehicles.

Roundabout lanes classification and priorities In accordance to the problem statement and the three rules listed in section 4.3, we propose to decompose a roundabout into three types of zones as illustrated in Figure 41. Each zone describes sub-steps of the insertion maneuver and a different priority rank, as follows:

a) The Decision Zone (in green in Fig. 41) is before the merging into the roundabout ring. In this zone, the AD vehicle does not have priority w.r.t. vehicles in the roundabout. It has to evaluate the possibility of a safe insertion in the roundabout without violating priority constraints.

b) The Transition Zone (in yellow in Fig. 41) is the last part of the entering lane where it merges with the roundabout ring. In this part, the AD vehicle performs a transition to enter into the roundabout. When the AD vehicle is in that zone, a safety inter-distance w.r.t. a potential incoming MD vehicle on the roundabout ring must be kept in order to allow a safe insertion.

c) The Ring Zone (in red in Fig. 41) corresponds to the roundabout ring. In this zone, the insertion maneuver is completed and the AD vehicle follows the nearest MD vehicle in the roundabout or drives at its nominal speed if it is alone.

d) The *Exit Zone* (in dark gray in Fig. 41) is the zone where the AD vehicle leaves the roundabout and continues its navigation following its path.

As a consequence, the crossing of the whole transition zone must be taken into account in the decision-making procedure. In fact, once a vehicle enters into that zone, it can no longer change its decision. If it needs to stop, it means that the decision of entering into the roundabout was wrongly taken which makes the approaching vehicle decelerate in order to avoid a collision. Finally, once the AD vehicle has crossed the transition zone, it gains the same priority as all the vehicles in the roundabout ring.

Intervals-based virtual platooning One well-known technique to achieve intersection crossing is to establish a crossing order between incoming vehicles [40]. However, when using intervals to represent the curvilinear occupancy of vehicles, cases where no total order between vehicles may arise if we apply virtual vehicle methods as in [40] and [1]. This is due to possible intervals overlapping.

Let us consider the case illustrated in Figure 43a, where the trajectory of the AD vehicle (in blue) crosses the one of an MD vehicle (in green) at a point n. Let us define $[\underline{s}, \overline{s}]$ the curvilinear occupancy of the AD vehicle w.r.t. the origin n, and similarly $[\underline{s}_i, \overline{s}_i]$ for an MD vehicle V_i. Figure 43b illustrates the six possible relative positions between the AD vehicle and the MD ones with their corresponding virtual projections (Fig. 43c).

Note that if there is no priority constraint between the vehicles, using intervals does not lead to a unique order among them.

Moreover, if there is no total order between the vehicles and no priority constraints, some deadlock situations may arise, as we previously discussed in [41].

To overcome this issue, one needs to choose an insertion policy [1]. This issue is out of the scope of this work, as we consider that vehicles inside the roundabout have the highest priority. In Figure 43c, one can see that in all the cases (2), (3),..., (6), the AD vehicle cannot be guaranteed to be in front of the incoming MD vehicle. These cases should not occur when the AD vehicle goes through the transition zone otherwise it will be in contradiction with rule R2. If the AD vehicle cannot guarantee that such cases will not occur, the AD vehicle decreases its speed to let the incoming vehicle go ahead. Such maneuver can lead either to a safe stop at the give-way line or to a speed adaptation depending on the relative intervals overlapping over time. In other words, the AD vehicle tries to adapt as much as possible its behavior to let the other car go first and if it cannot, it performs a safe stop. Notice that the only case where the AD vehicle may expect to be in front of

the MD vehicle is the case (1).

Let us consider the scenario depicted in Figure 44, where the AD vehicle V_0 is in the decision zone of the roundabout and an incoming priority MD vehicle V_1 is in the ring zone. This scenario falls clearly into case (1), with $d^*_{0,1} > 0$.

Let us define t_0 as the time when the front part of the AD vehicle enters into the transition zone. At a given time $t < t_0$ (*i.e.,* when the AD vehicle was still in the decision zone), one can see that having

$$d^*_{0,1}(t_0) > d_{\text{safe}}, \tag{11}$$

is not sufficient for the AD vehicle to ensure a safe and priority preserving insertion maneuver according to rule R2. Indeed, if the speed v_1 of vehicle V_1 is greater than the speed v_0 of V_0, the inter-distance between the two vehicles will shrink over time. This shows that vehicle kinematics must be taken into account at the decision-making level.

Considering again rule R2, the AD vehicle needs to guarantee that equation (11) will be satisfied during the whole insertion maneuver, *i.e.,* from the moment that the upper bound $\overline{s}$ enters the transition zone until the lower bound $\underline{s}$ leaves it. Let Δt be the time needed by the AD vehicle to go completely through the transition zone, *i.e.,* the back of the AD vehicle has exited it. The decision to enter the roundabout is taken if:

$$\forall t \in [t_0, t_0 + \Delta t], \quad d^*_{0,1}(t) > d_{\text{safe}}. \tag{12}$$

In order to guarantee the inequality of equation (12), we need to know how both $\underline{s}_{0|n}$ and $\overline{s}_{1|n}$ evolve over time. In this work, we assume that both vehicles drive at a constant speed. This assumption may seem simplistic, but it is representative of the driver behavior as indicated in the INTERACTION dataset. Indeed, within the roundabout ring of the INTERACTION dataset, the speed profiles of the MD vehicles have a standard deviation less than 1 m/s in average. Under this assumption, we have $\Delta t = l/v_0$, where l is the length of the transition zone, and the kinematics of each interval can be expressed as follows (bearing in mind the aforementioned considerations):

$$\underline{s}_{0|n}(t) = \underline{s}_{0|n}(t_0) - v_0 \cdot (t - t_0), \tag{13}$$

$$\overline{s}_{1|n}(t) = \overline{s}_{1|n}(t_0) - v_1 \cdot (t - t_0). \tag{14}$$

Substituting equations (13) and (14) in (12), we obtain

$$\underbrace{\overline{s}_{1|n}(t_0) - \underline{s}_{0|n}(t_0)}_{=d^*_{0,1}(t_0)} + (v_0 - v_1) \cdot (t - t_0) \geq d_{\text{safe}}, \tag{15}$$

The inequality (15) needs to hold $\forall t \in [t_0, t_0 + \Delta t]$.

If $v_0 > v_1$, it leads to

$$d^*_{0,1}(t_0) \geq d_{\text{safe}}. \tag{16}$$

It means that if the AD vehicle drives faster than V_1, it can insert if it is sufficiently ahead of V_1 at t_0.

Otherwise, if $v_0 < v_1$, we have

$$d^*_{0,1}(t_0) \geq d_{\text{safe}} + \left(\frac{v_1}{v_0} - 1\right) l. \tag{17}$$

One can see that the relative speed v_1/v_0 needs to be taken into account in the decision and that if equation (17) holds at t_0, it also holds for all the interval $[t_0,\, t_0+\Delta t]$. This is particularly useful in the case where the AD vehicle accelerates from a low speed ($v_0 = 0$ in the case of a stop at give way) to enter into the roundabout because it already encompasses all the speed changes of the AD vehicle in the decision. In other words, the decision taken at t_0 cannot change if the speed of the AD vehicle increases.

Inequality (17) allows the AD vehicle to decide if it has enough space to keep a safety inter-distance w.r.t. an eventual incoming vehicle, knowing its velocity and its occupancy at time t_0. In the case where equation (17) is not satisfied, the ego vehicle slows down to perform a safe stop at the end of the decision zone (that coincides with the give way marking).

Nevertheless, once the speed of the AD vehicle is close to zero, it is difficult for the AD vehicle to find a sufficiently large gap to perform the insertion. This is due to the singularity present in equation (17) when $v_0 = 0$. In fact, considering the function $h(v_0) = \left(\frac{v_1}{v_0} - 1\right)$, one can see that it has a peak towards $+\infty$ for $v_0 \to 0$. This degrades the performance of the algorithm once the AD has stopped: it will only insert once no incoming vehicle is present. To overcome this, we propose to replace $h(v_0)$ with another function for the case $v_0 < v_1$. In particular, we look for a function that meets the following criteria:

1. As $v_0 \to 0$, the value of the function becomes less dependent on v_0.
2. For $v_0 = 0$ the value of the function depends at least on v_1.

The main idea is to have a function that allows to set a safety gap that depends at least only on the other vehicle speed v_1. This solution is convenient when the dynamic of the system is not well known and we need to perform a prediction without being too pessimist. In this work, we choose a function with the following form:

$$\widehat{h}(v_0) = A\left(\frac{1}{2} - \frac{1}{1 + e^{-\alpha(v_0 - v_1)}}\right) \tag{18}$$

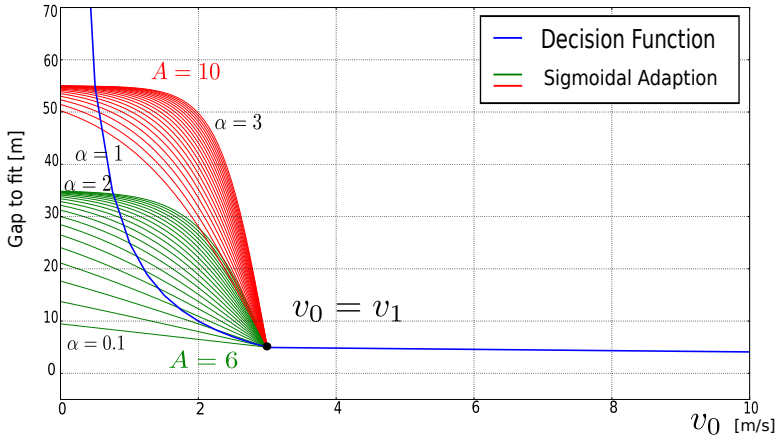


Figure 45: The behavior of the decision function with h and $\widehat{h}$ (19) in terms of required inter-distance for several values of A and α for a fixed value of v_1, l and d_{safe}.

Where the two parameters A and α have to be tuned experimentally to make a good insertion maneuver (see section 4.3). Equation (17) now becomes

$$\begin{cases} d^*_{0,1}(t_0) \geq d_{\text{safe}} & \text{if } v_0 > v_1, \\ d^*_{0,1}(t_0) \geq d_{\text{safe}} + \widehat{h}(v_0)l & \text{else.} \end{cases} \tag{19}$$

or equivalently

$$\begin{cases} d^*_{0,1}(t_0) - d_{\text{safe}} \geq 0 & \text{if } v_0 > v_1, \\ d^*_{0,1}(t_0) - d_{\text{safe}} - \widehat{h}(v_0)l \geq 0 & \text{else.} \end{cases} \tag{20}$$

Figure 45 illustrates the function $\widehat{h}$ for several values of A and α. As one can see, the singularity present in h for $v_0 = 0$ is avoided. For instance, for $\alpha = 1$ and $A = 10$, if the AD vehicle is at $v_0 = 0$, then it decides to enter if the inter-distance is around 50 m. To describe how to apply the aforementioned concepts in the case of a traffic flow, algorithm .1 details the insertion strategy in a general case. In such case, one can notice that when the AD vehicle has at least one vehicle behind and one ahead, the prediction (Eq. 20) is done considering the vehicle ahead speed instead of the AD one. This allows to take into account the presence of an eventual vehicle ahead during the insertion maneuver. On the other hand, when there is at least one vehicle behind the AD car that does not satisfy Eq. 20, the AD vehicle performs a safe stop at the give way, and it chooses the vehicle to follow for entering into the roundabout as the farthest object that does not satisfy Eq. 20 in order to avoid unsafe configurations.

Algorithm .1 Roundabout insertion for an AD vehicle V_0.

Require: V_0, t_0

$[V_1,\ V_2,\ \ldots,\ V_c] \leftarrow$ Perception() ▷ Other road users
$\Delta t \leftarrow l/v_0$
$leader \leftarrow \emptyset, \quad d_{leader} \leftarrow -\infty$
$V_{RISK} \leftarrow []$ ▷ List of vehicles that represent a risk
for $j = 1 : c$ **do**
    $n \leftarrow \text{FindFirstCommonNode}(P_0, P_j)$
    if $n = \emptyset$ **then**
        Continue ▷ V_j does not cross the path of V_0
    else
        $d^*_{0,j|n}(t_0) \leftarrow \overline{s}_{j|n}(t_0) - \underline{s}_{0|n}(t_0)$
        if $d^*_{0,j|n}(t_0) < 0$ **then** ▷ Vehicle V_j in front of V_0
            if $d^*_{0,j|n}(t_0) > d_{leader}$ **then**
                $d_{leader} \leftarrow d^*_{0,j|n}(t_0)$
                $leader \leftarrow V_j$
            else
                Continue
            end if
        else ▷ Vehicle V_j behind or intersecting V_0
            if $d^*_{0,j}(t_0) \geq d_{\text{safe}} + \left(\frac{v_j}{v_0} - 1\right) l$ **then**
                Continue ▷ Contraint satisfied
            else
                $V_{RISK} \leftarrow V_{RISK} \cup [V_j, d^*_{0,j}(t_0)]$
            end if
        end if
    end if
end for
if $leader = \emptyset$ **then**
    $v_d \leftarrow v_n$ ▷ Go with nominal speed
else
    $v_d \leftarrow leader.speed$ ▷ Speed of vehicle ahead
end if
if $V_{RISK} \neq []$ **then** ▷ Change leader vehicle
    $leader \leftarrow \max_{d^*_{0,j|n}} [V_{RISK}]$
    $v_d \leftarrow leader.speed$
end if
SetSpeed(v_d) ▷ Perform longitudinal control

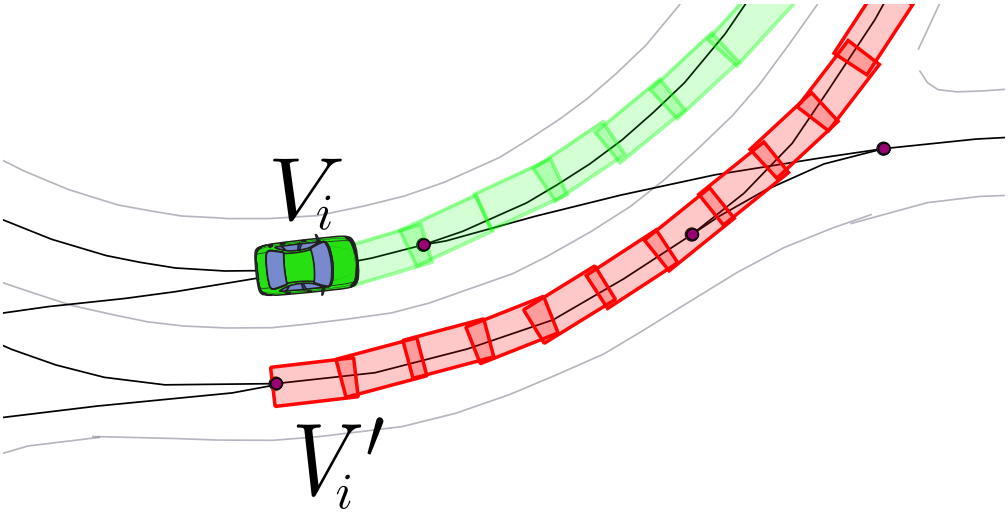


Figure 46: Illustration of the strategy to handle the lane change maneuvers in a two-lane roundabout. The vehicle trajectory is green and the corresponding lane occupation is red.

Simulation results To validate our approach, we have used two simulators. The SUMO simulator [43] was used for microscopic traffic flow generation while a ROS-based simulation framework was used to implement the navigation algorithm and the AD vehicle dynamics. The coupling and synchronization of both simulators have been achieved with the TraCi SUMO library and its Python API. A detailed explanation on time synchronization and coupling of the two simulators can be found in [44].

Furthermore, we have imported in the SUMO simulation environment the HD map representation of the test-bed roundabout. This has been achieved with the Netedit and Netconvert tools included in the SUMO suite [45]. Figure 47 pictures an overview of the roundabout scenario in both simulators.

For each simulation, a random high density vehicle flow that meets the $\Delta TTIC_{min}$ criterion has been generated over a fixed time horizon $T = 200$ s. Each simulation contained between 200 and 400 seconds for a total amount of more than 1 hour of simulation. The number of vehicles for each flow randomly varied between 50 and 175, for a total amount of more than $5,000$ vehicles.

These limits have been chosen to capture a wide range of scenarios, starting from a sparse traffic flow until a denser vehicle stream. Moreover, to simulate both localization (AD vehicle) and perception (MD vehicles) uncertainty, we consider a ± 1 m bound to add to the curvilinear interval $[\underline{s}_i, \overline{s}_i]$ and $[\underline{s}, \overline{s}]$, which represents the projection of the vehicle footprint on its lane.

To experimentally validate our approach, we consider several scenarios with the

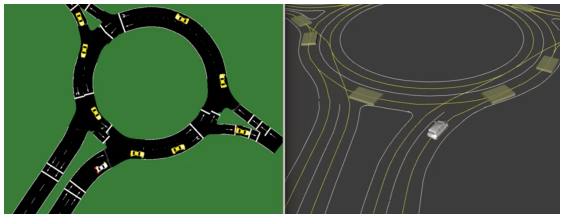


Figure 47: The coupled simulator. The AD vehicle is in gray, while all the MD vehicles are in yellow. Note that the roundabout in SUMO has been designed using the HD map representation of the roundabout shown in Fig. 41.

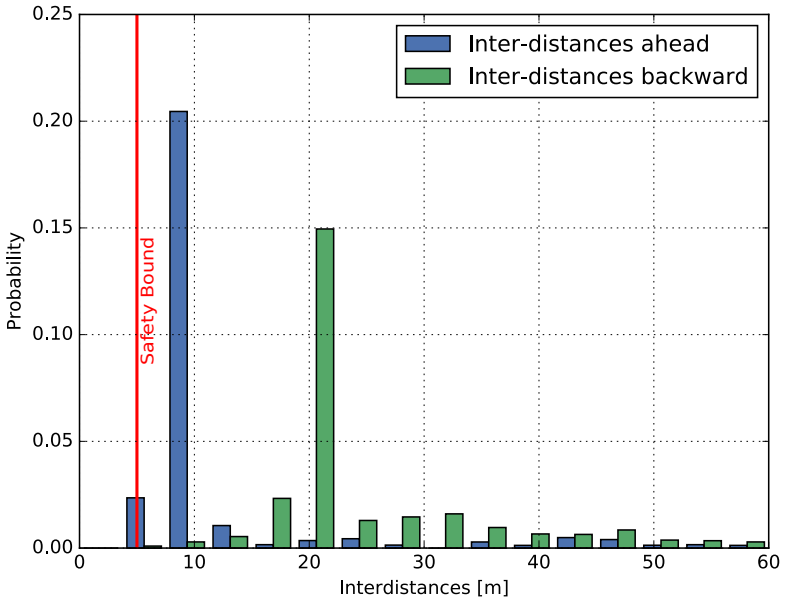
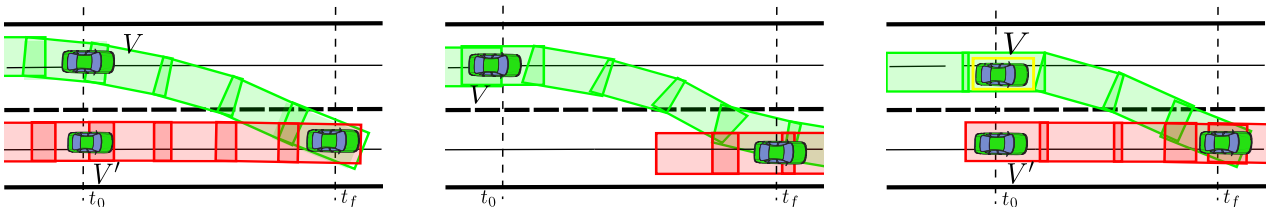


Figure 48: Inter-distance distributions w.r.t. the vehicle ahead (blue) and behind (green) during an insertion maneuver. The red line represents the 5 m safety gap.

Table 1: The average insertion time, the percentage of average waiting time relative to the nominal case and the average number of insertions as function of vehicles flow for the single-lane roundabout case.

Flow size	50	75	100	125
Crossing Time (s)	5.60 (+1.3)	7.32 (+1.7)	10.05 (+2.39)	15.26 (+3.63)
Number of Insertions	24	21	18	16



(a) (b) (c)

Figure 49: Illustration of the three strategies to handle the lane change maneuvers in a two-lane roundabout. The vehicle trajectory is green and the corresponding lane occupation is red. Note that for methods 49c the lane occupation becomes red when the intention to change lane is detected (yellow square)

aforementioned technique to generate high density traffic flows. The simulations have been carried out with the HD map representation of the roundabout displayed on Figure 41.

To better quantify the performance of our algorithm, we have computed the distributions of the inter-distances w.r.t. the vehicle ahead and behind during the crossing of the transition zone. Figure 48 shows the inter-distance distributions for both the ahead and behind gaps. As one can see, the behind safety gap always meets the safety criterion. Conversely, considering the gap w.r.t. vehicles ahead, there is a slight violation of the safety bound. This violation is due to some controller imperfection and can be neglected.

Moreover, we are interested in quantifying the insertion rate w.r.t. the vehicles flow. Table 1 illustrates the average insertion rate and waiting time as a function of the number of vehicles in the flow. The insertion time is computed considering that the total length of the decision and transition zones is 33.4 m and the nominal speed in this zone is 30 km/h, which gives us a nominal waiting time of 4.2 seconds.

As one can see, the number of insertions decreases w.r.t. the number of vehicles in the flow. Conversely, the waiting time increases.

4.4 Two-lane Roundabout Crossing

Problem statement Let us see in this section how to exploit the virtual vehicles to model the behavior of other vehicles to cope with nudging behaviors. We consider again the roundabout shown in Figure 41. Remember that according to our navigation strategy, an AD vehicle crosses a roundabout only by following the outermost lane (and therefore cannot change lanes). One of the most difficult issues in this scenario is the handling of vehicles that perform lane change maneuvers from the inner ring of the roundabout to outer one which is the one that the AD vehicle uses in our method. In practice, it is very challenging to predict a lane change maneuver, especially when vehicles attempt to make a lane change with nudging [46]. We propose three different strategies to handle the navigation of multiple vehicles inside a two-lane roundabout.

To do that, we use again the concept of virtual vehicles as we did before to predict the intention of other vehicles (see Fig. 44). In particular, the main idea is to generate an extra virtual instance of a given vehicle on the innermost lane of the roundabout to occupy the outermost lane according to occupancy methods.

The first method consists in occupying systematically both lanes of the roundabout ring if at least one lane is occupied. This means that, if a vehicle is occupying the innermost lane, the outermost one will result occupied too. Fig. 49a illustrates this method. The second method consists in occupying the lanes only when there is a significant physical occupancy of a vehicle. This implies that during a lane change maneuver there is always one occupied lane at most, *i.e.,* the outer lane becomes occupied only when the first half of the vehicle has already crossed the bound between the two lanes of the roundabout (Fig. 49b). The third method occupies the outer lane only when the intention of a vehicle to change lane is detected. This approach is shown in Fig. 49c. In this case, we assume to have a system that is able to predict when a driver decides to change lane, *e.g.*, by detecting blinkers or lateral distance from the lane center.

Summarizing, the three aforementioned methods [38] generate an extra virtual instance according to the following statements:

1. Occupying systematically both lanes of the roundabout ring if at least one lane is occupied.

2. Occupying the lanes only when there is a significant physical occupancy of a vehicle.

3. Occupying the outer lane only when the intention of a vehicle to change lane is detected.

These three approaches have been compared in terms of both safety and system availability and the results are reported in section 4.4. However, in order to use proficiently such method, a precise lane change intention detector is required. In

Table 2: Probability of a safety bound violation (ahead and behind), crossing times and effective waiting times for the three methods.

Method	behind	Ahead	Cross time	Wait time
Two lanes occupancy	0%	0%	23.3s	14.46s
Only one lane occupancy	30%	29%	8.6s	4.7s
Intention detection	0%	4%	16.4s	7.79s

this work, we limit our approach to the method based on statement 1 (Fig. 46) because such lane change intention predictor is not available in our system architecture and our roundabout test bed is composed of only a one-lane roundabout.

Simulation results For every strategy listed hereafter, several simulations on SUMO with a randomly generated vehicle flow on the full roundabout (two lanes) have been carried out. To properly model a lane change maneuver inside SUMO, we consider a nonzero duration for lane changes and the other hypotheses done in [45]. In order to quantify the performance of our approach, Fig. 50 and 51 show respectively the inter-distances w.r.t. the vehicles ahead and behind for the three strategies. The first method always ensures safety. For the other two, it is not the case. In fact, the third method violates the safety constraints in both forward and backward cases. This is due to the late detection of lane changes. As a consequence of that, it can happen that the ego vehicle performs an insertion maneuver when another vehicle has already decided to change lane. In this situation, two scenarios can happen: The ego vehicles force the other car to change its intentions or the other car complete the lane change despite the ego vehicle presence. In the first case, this driving behavior is called nudging . In the second case, the other vehicle cut off the road of the ego vehicle, resulting in a hazardous maneuver or on a collision. The third case behaves as a compromise between the two.

To analyze this aspect, Table 2 shows the collision probabilities, average insertion times and average waiting times for the three strategies. Logically, if one wants to have safety ensured, the average waiting time increases. This is due to the fact that, with the aforementioned methods for a double lane roundabout, the available spaces slots where the AD vehicle can insert are reduced because, in two cases at least, a vehicle matched in the innermost lane produces also an occupancy on the outermost one. As a consequence, the capacity on the roundabout ring is more reduced w.r.t. a single lane one. As a consequence, the first method (Fig. 49a) tends to be too overly-conservative. Conversely, the second method (Fig. 49b) is much more aggressive because of the lack of lane change prediction. However, this method has a large unsafe set of configurations. Finally, the third method (Fig. 49c) behaves as a compromise in-between.

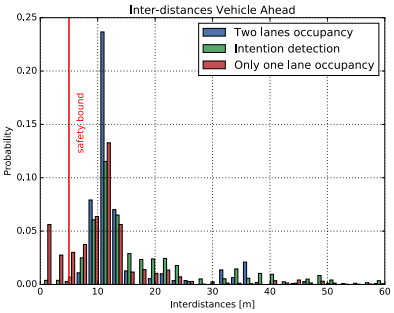


Figure 50: The inter-distances distributions w.r.t. the vehicle ahead for the three strategies presented in section 4.4.

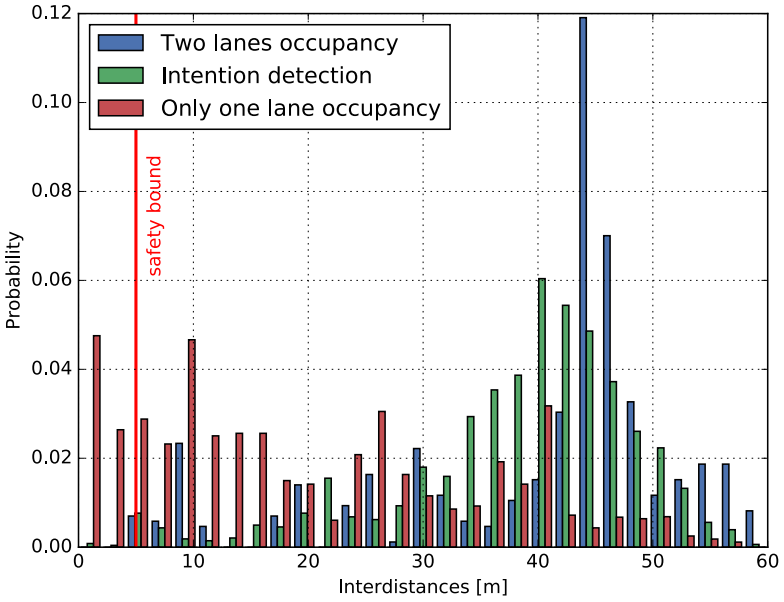


Figure 51: Inter-distances distributions w.r.t. the vehicle behind for the three strategies. One can see that the intention detection method behaves as a compromise between the other two.

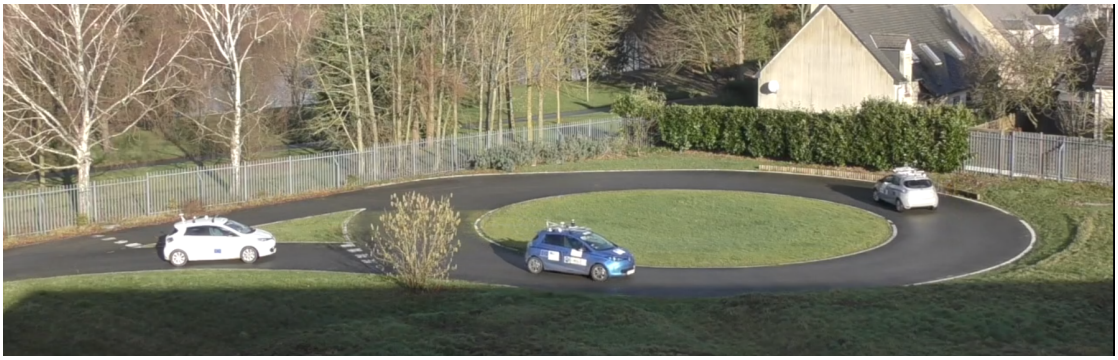


Figure 52: The experimental circuit “Seville” and the experimental Renault ZoÃ©s used during the real outdoor tests. In our configuration, the white car (fully autonomous) is the AD vehicle, while the others are the road users (manually driven).

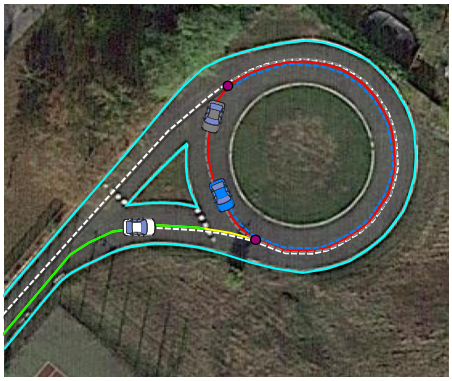
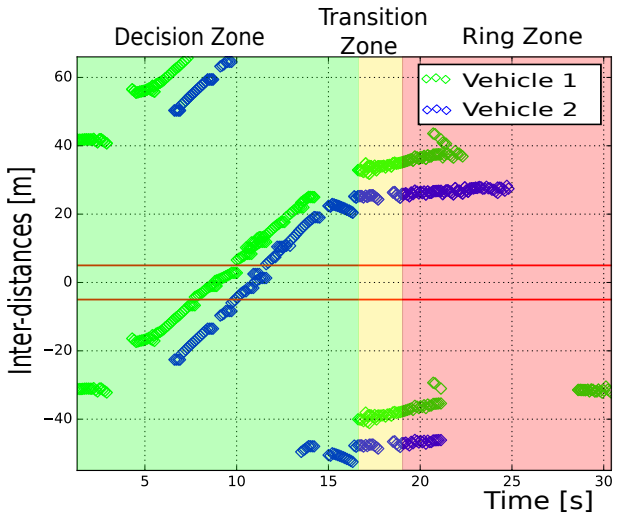


Figure 53: The experimental circuit “Seville” with the corresponding HD map and the experiment configuration depicted. The AD vehicle is blue and the MD vehicles are green. Notice that the situation case described in section 4.3 may occur.

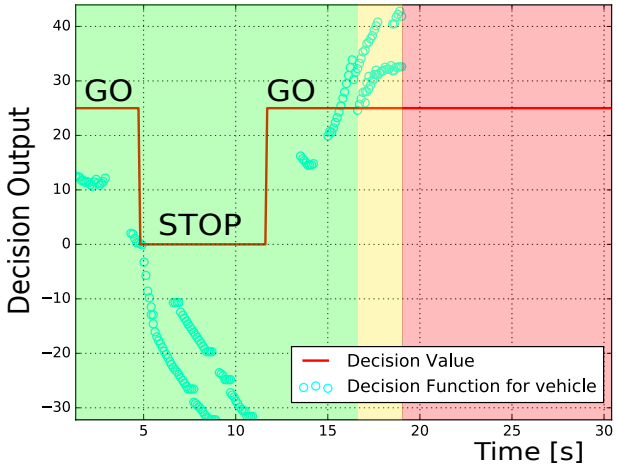
4.5 Real experiments

In order to validate our strategy in a real scenario, we have implemented the whole system architecture on an autonomous Renault ZoÃ© using the ROS middleware. More details about the system architecture of our experimental cars are provided in [47].

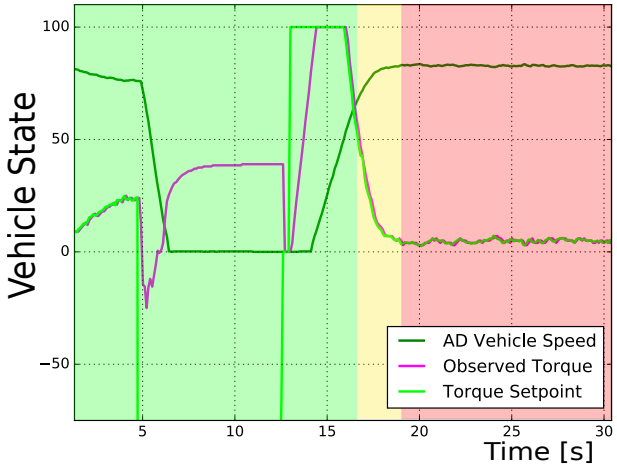
We have tested the roundabout insertion first in a hybrid environment (*i.e.*, with simulated vehicles moving on the test track) then with real road agents detected with a LiDAR-based perception system. In the experimental autonomous car, the system controls the throttle, the brake pedal and the steering wheel. In our case, the longitudinal motion of the vehicle (*i.e.*, acceleration and brake) is computed according to the traffic situation. In particular, based on the algorithm explained in section 4.3, the vehicle can either perform an insertion maneuver into the roundabout or decrease its speed to let other cars go ahead, eventually effec-



(a)



(b)



(c)

Figure 54: Experimental results. Fig.54a illustrates the (virtual) inter-distances of the two cars w.r.t the AD vehicle as a function of time. Notice that the same vehicle may appear both behind and in front of the AD vehicle because of the vehicle instance concept. Figure 54b depicts the value of the decision function (red) and the result of equation 20 (blue dots) for every vehicle (virtually) behind the AD vehicle (points < 0 in Fig. 54a). Figure 54c shows the torque setpoint and the corresponding vehicle torque and speed (amplified by a scale factor) as a function of the decision taken in Fig. 54b.

tuating a stop at the give-way road sign. Regarding lateral motion, a simple lane keeping is performed and for the detection part, we use a state-of-the-art LiDAR object detection algorithm [42] able to provide information about the detected objects in the form $V_i = [\underline{s}_i, \overline{s}_i, v_i, P_i]$. Note that P_i and the curvilinear conversion have been computed according to previously explained methods.

Let us consider a scenario where two cars drive close to each other inside a roundabout doing infinite loops and the AD vehicle has to enter the roundabout (Figure 53). Figure 54a illustrates the (virtual) inter-distances of the AD vehicle w.r.t. the other road agents (backward and ahead) accordingly to the three zones. As we can see, the sign of such distance depends on the relative positioning between the MD vehicles and the AD vehicle.

Notice that due to the concept of vehicle instances a vehicle inside the roundabout can be both ahead and behind the AD vehicle, which is coherent with the circular shape of the roundabout. Furthermore, the colors of the background denote the zone of the roundabout where the AD vehicle is. As one can see, during the crossing of the transition and ring zones, the safety gap is always kept.

If we observe carefully the points in the decision zone, one can see that the other road agents virtually overtake the AD vehicle. In fact, Figure 54b shows that, due to the negative values of the constraint of equation (19), the decision changes from go to stop. As a consequence, the AD vehicle lets the other vehicle go ahead and enters into the roundabout behind them.

To better understand the situation at decision and control level, Figure 54c shows the system behavior during the whole maneuver. In particular, we can see that the set-point torque changes following the decision output in Figure54b. Consequently, the vehicle speed and the applied torque to the engine change accordingly.

Finally, when the red function decreases to zero in Figure 54b, the controller performs a safe stop maneuver. Conversely, once the decision-making part decides to let the vehicle enter in the roundabout, the controller accelerates accordingly. Note that when the vehicle is completely stopped (*i.e.*, $v = 0$) it is still in the decision zone. This means that no safety violation occurred during the maneuver.

4.6 Conclusion

In this document, we have studied an adaptation of the virtual platooning concept to the roundabout crossing problem. This idea has the advantage to be easily implemented in an embedded system that exploits a map-based approach. This work has also shown the importance of exploiting a map to model a roundabout since all the calculations are done in a curvilinear framework.

Then, an extension of this strategy to real-word traffic has been proposed to handle the uncertainty of localization and the issues linked to the detection and

tracking of uncertain objects. Furthermore, uncertainties in regular vehicles intentions are hard to predict and understand too. Finally, traffic regulations, such as the right of way, have been added to the framework to make this approach effective in a real-life scenario. First, we have presented how the virtual instances of vehicles can be used to handle not only the MD vehicles unknown intentions, but also the particular shape of roundabouts. Then, an approach with occupancy intervals to calculate the best gap to fit during a roundabout insertion maneuver has been proposed. The choice of representing objects occupancy with intervals has been made to better include the size of MD vehicles and a possible uncertainty about their estimated occupancies. This approach has been tested under a simulated traffic flow generated from real data. The degree of interaction of the generated flow has been used to re-create a scenario close to real world driving. We have shown that the proposed insertion maneuver ensures safety. We have also proposed some performance indexes to evaluate its efficiency in terms of traffic fluidity.

To handle the problem of a double-lane roundabout, a lane change intention detector is required to obtain safe and not overly cautious performance. If this technology is unavailable, we suggest using instead a worst-case occupation method that always provides a safe insertion.

In facts, based on the obtained results, it is our opinion that, in order to safely cross a roundabout ensuring safety and without being too overly conservative, an accurate lane change intention detection algorithm is required.

Moreover, if the lane change detector is able to detect also nudging, one can discriminate between a real intention of a driver to effectuate a lane change and a false alarm. This approach should add efficiency in the insertion maneuver, decreasing waiting times without compromising safety.

Finally, the method has been tested on an experimental test circuit with real road users and a real AD vehicle, in order to evaluate the performance of the proposed algorithm in a real-world scenario with a perception system that provides information about the surrounding road agents.

As a future perspective, one could extend the tests with a real vehicle to more complex scenarios (e.g., two-lane roundabouts with more MD vehicles) and to include in their architecture a lane change intention detector. Furthermore, in order to obtain a better performance when the traffic flow is dense, it would be interesting to study how a negotiation layer involving intentions and actions of other parties during the interaction could be implemented. This layer can play an important role when AD vehicles are involved in a decision process.

References

[1] S. Masi, P. Xu, and P. Bonnifait, "Adapting the virtual platooning concept to roundabout crossing," in IEEE Intelligent Vehicles Symposium, 06 2018, pp. 1366–1372.

[2] I. Besselink, S. Achrifi, and H. Nijmeijer, "Lateral vehicle dynamics on rutted roads," in The IAVSD International Symposium on Dynamics of Vehicles on Roads and Tracks. Springer, 2019, pp. 1242–1251.

[3] L. Xiao and F. Gao, "A comprehensive review of the development of adaptive cruise control systems," Vehicle system dynamics, vol. 48, no. 10, pp. 1167–1192, 2010.

[4] T. Luettel, M. Himmelsbach, and H.-J. Wuensche, "Autonomous ground vehicles concepts and a path to the future," Proceedings of the IEEE, vol. 100, no. Special Centennial Issue, pp. 1831–1839, 2012.

[5] A. Broggi, P. Cerri, S. Debattisti, M. Laghi, and P. Medici, "Proud–public road urban driverless-car test," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 6, pp. 3508 – 3519, 2015.

[6] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller et al., "Making bertha drive an autonomous journey on a historic route," IEEE Intelligent transportation systems magazine, vol. 6, no. 2, pp. 8–20, 2014.

[7] A. Soteropoulos, M. Berger, and F. Ciari, "Impacts of automated vehicles on travel behaviour and land use: an international review of modelling studies," Transport reviews, vol. 39, no. 1, pp. 29–49, 2019.

[8] C. Chmielewski, "Self-driving cars and rural areas: The potential for a symbiotic relationship," Journal of Law & Commerce, vol. 37, no. 1, pp. 57 – 81, 2019.

[9] P. Hammond, "2007-2026 highway system plan: High benefit low cost," Washington State Department of Transportation, Tech. Rep., December 2007.

[10] "DÃ©cret nÂ°2019-1082 du 23 octobre 2019 - art. 2: Article r110-2," Securite Routiere France, Tech. Rep., October 2019.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in Proc. IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980–2988.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[14] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2117–2125.

[15] C. Guindel, J. Beltrán, D. Martín, and F. García, "Automatic extrinsic calibration for lidar-stereo vehicle sensor setups," in Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 674–679.

[16] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 918–927.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 652–660.

[18] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in Proc. IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium, 2000, pp. 153–158.

[19] F. Li, "Lane-level vehicle localization with integrity monitoring for data aggregation," Ph.D. dissertation, Compiègne, 2018.

[20] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in 2015 IEEE 18th International Conference on Intelligent Transportation Systems. IEEE, 2015, pp. 982–988.

[21] D. Jiang and L. Delgrossi, "Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments," in VTC Spring 2008-IEEE Vehicular Technology Conference. IEEE, 2008, pp. 2036–2040.

[22] "Intelligent transport systems cooperative its using v2i and i2v communications for applications related to signalized intersections," International Organization for Standardization, Geneva, CH, Standard, Jun. 2019.

[23] A. Piazzi, C. L. Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, "Quintic gsup 2-splines for the iterative steering of vision-based autonomous vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 3, no. 1, pp. 27–36, 2002.

[24] E. W. Dijkstra et al., "A note on two problems in connexion with graphs," Numerische mathematik, vol. 1, no. 1, pp. 269–271, 1959.

[25] V. Milanes, F. Navas, D. Gonzalez, and I. Mahtout, "On the passenger acceptance of driverless shuttles," IEEE Intelligent Transportation Systems Magazine, InPress 2020.

[26] H.-S. Tan and J. Huang, "Design of a high-performance automatic steering controller for bus revenue service based on how drivers steer," IEEE Transactions on Robotics, vol. 30, no. 5, pp. 1137–1147, 2014.

[27] H. Niemann and J. Stoustrup, "An architecture for implementation of multivariable controllers," in American Control Conference, 1999. Proceedings of the 1999, vol. 6. IEEE, 1999, pp. 4029–4033.

[28] K. Zhou, J. Doyle, and K. Glover., Robust and Optimal Control, P. Hall, Ed., 1996.

[29] I. Mahtout, F. Navas, D. Gonzalez, V. Milanes, and F. Nashashibi, "Youla-kucera based lateral controller for autonomous vehicle," in 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018, pp. 3281–3286.

[30] F. Wu, Control of linear parameter varying systems. University of California, Berkeley, 1995.

[31] P. Apkarian and R. J. Adams, "Advanced gain-scheduling techniques for uncertain systems," in Advances in linear matrix inequality methods in control. SIAM, 2000, pp. 209–228.

[32] J. Lussereau, P. Stein, J.-A. David, L. Rummelhard, A. Negre, C. Laugier, N. Vignard, and G. Othmezouri, "Integration of adas algorithm in a vehicle prototype," in 2015 IEEE International Workshop on Advanced Robotics and its Social Impacts, 2015.

[33] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm." in 3dim, vol. 1, 2001, pp. 145–152.

[34] M. E. El Najjar and P. Bonnifait, "A road-matching method for precise vehicle localization using belief theory and kalman filtering," Autonomous Robots, vol. 19, no. 2, pp. 173–191, 2005.

[35] C. A. Blazquez, J. Ries, and P. A. Miranda, "Towards a parameter tuning approach for a map-matching algorithm," in 2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES). IEEE, 2017, pp. 85–90.

[36] R. Asghar, M. Garzon, J. Lussereau, and C. Laugier, "Vehicle localization based on visual lane marking and topological map matching," 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 258–264, 2020.

[37] L. Rummelhard, A. Nègre, and C. Laugier, "Conditional monte carlo dense occupancy tracker," in 2015 IEEE 18th International Conference on Intelligent Transportation Systems. IEEE, 2015, pp. 2485–2490.

[38] S. Masi, P. Xu, and P. Bonnifait, "A curvilinear decision method for two-lane roundabout crossing and its validation under realistic traffic flow," in IEEE Intelligent Vehicles Symposium, 06 2020, p. in proceedings.

[39] E. Hery, S. Masi, P. Xu, and P. Bonnifait, "Map-based curvilinear coordinates for autonomous vehicles," in IEEE 20th International Conference on Intelligent Transportation Systems, 2017.

[40] A. I. M. Medina, N. V. D. Wouw, and H. Nijmeijer, "Automation of a t-intersection using virtual platoons of cooperative autonomous vehicles," in IEEE 18th International Conference on Intelligent Transportation Systems, 09 2015, pp. 1696–1701.

[41] S. Masi, P. Xu, and P. Bonnifait, "Roundabout crossing with interval occupancy and virtual instances of road users," IEEE Transactions on Intelligent Transportation Systems (T-ITS), 11 2020.

[42] E. Bernardi, S. Masi, P. Xu, and P. Bonnifait, "High integrity efficient lane-level occupancy estimation of road obstacles through lidar and hd map data fusion," in IEEE Intelligent Vehicles Symposium, 06 2020, p. in proceedings.

[43] D. Krajzewicz, G. Hertkorn, C. Feld, and P. Wagner, "Sumo (simulation of urban mobility); an open-source traffic simulation," in 4th Middle East Symposium on Simulation and Modelling (MESM2002), 01 2002, pp. 183–187.

[44] M. Garzon and A. Spalanzani, "An hybrid simulation tool for autonomous cars in very high traffic scenarios," in 15th International Conference on Control Automation Robotics and Vision, 11 2018, pp. 803–808.

[45] M. Klischat, O. Dragoi, M. Eissa, , and M. Althoff, "Coupling sumo with a motion planning framework for automated vehicles," in SUMO User Conference, 2019, pp. 1–9.

[46] K. Leung, E. Schmerling, M. Chen, J. Talbot, J. C. Gerdes, and M. Pavone, "On infusing reachability-based safety assurance within probabilistic planning frameworks for human-robot vehicle interactions," in International Symposium on Experimental Robotics, pp.561-574, 2018.

[47] P. Xu, G. Dherbomez, E. Hery, A. Abidli, and P. Bonnifait, "System architecture of a driverless electric car in the grand cooperative driving challenge," IEEE Intelligent Transportation Systems Magazine (ITS Mag.), pp. Vol. 10, Issue 1, pages 47–59, 03 2018.