# System Architecture for Autonomous Driving on Route du Bray

Rabbia Asghar, Christian Laugier

# System Architecture for Autonomous Driving on Route du Bray

# Report for Project Tornado

Rabbia Asghar, Christian Laugier[1]

February 12, 2021

[1] Univ. Grenoble Alpes, Inria, 38000 Grenoble, France, email: FirstName.LastName@inria.fr

# Contents

# List of Figures

# Chapter 1

# Introduction

The TORNADO project is a multi-partner project, subsidized by the Fonds Unique Interministriel (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 1.1: 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 1.2 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

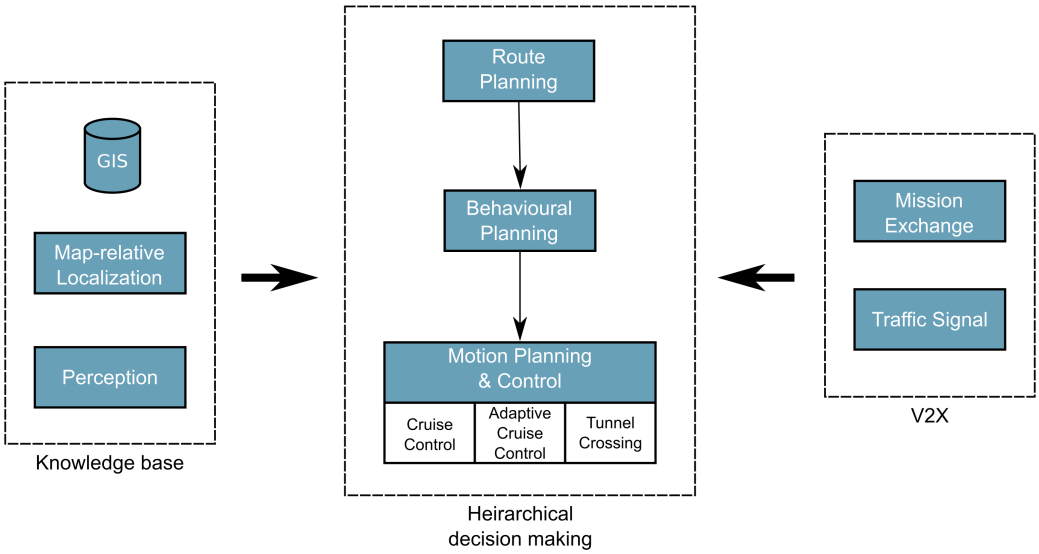



Figure 1.2: System architecture for autonomous driving on Route du Bray.

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.

# Chapter 2

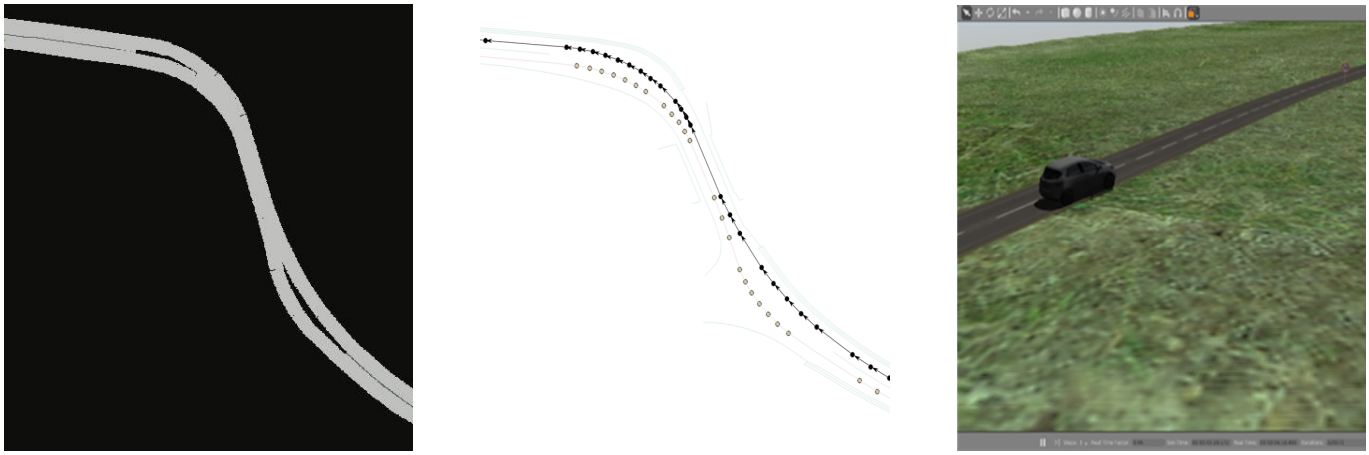
# Knowledge Base

## 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. 2.1a 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 2.1: 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.2.1b 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. 2.1c. 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.

## 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 [1], [2]. 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 [3], [4], [5], [6].

The map relative localization approach developed is based on Extended Kalman filtering (EKF), as illustrated in Fig. 2.2. 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.

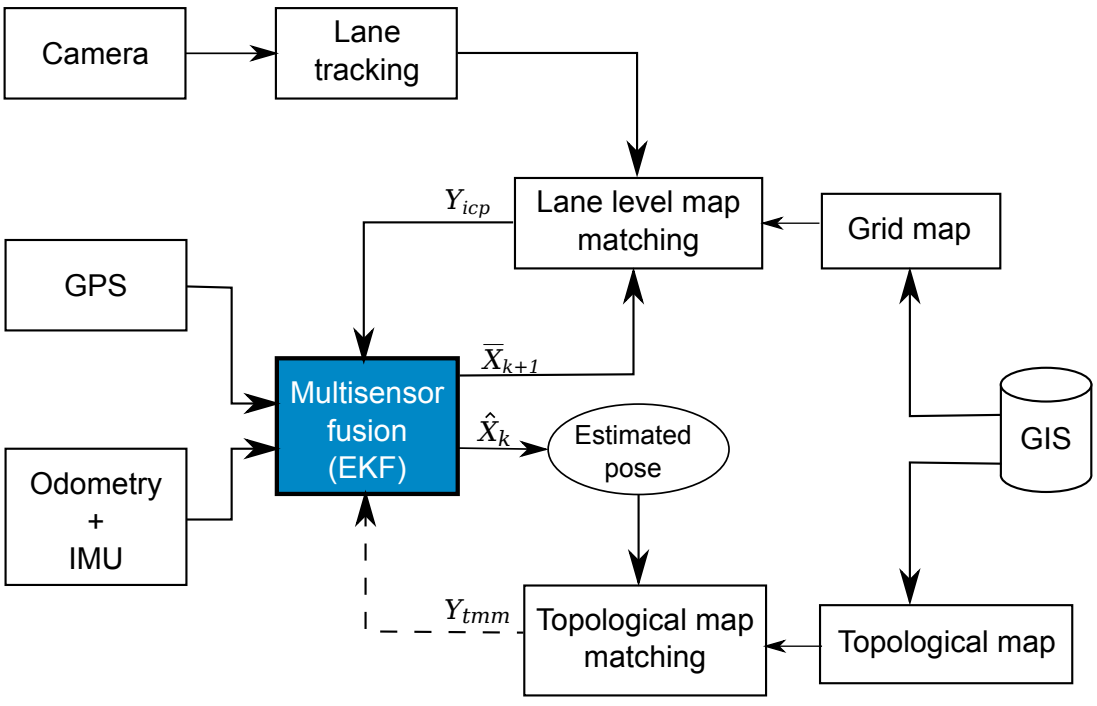



Figure 2.2: Overview of the map relative localization approach.

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 $\theta$. GPS measurements are converted to

ENU coordinates in local navigation frame (discussed previously in section 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. 2.2 are discussed in the sections 2.2.1 and 2.2.2 respectively.

### 2.2.1 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 [7] 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.2.3 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.2.3 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 [8] 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.

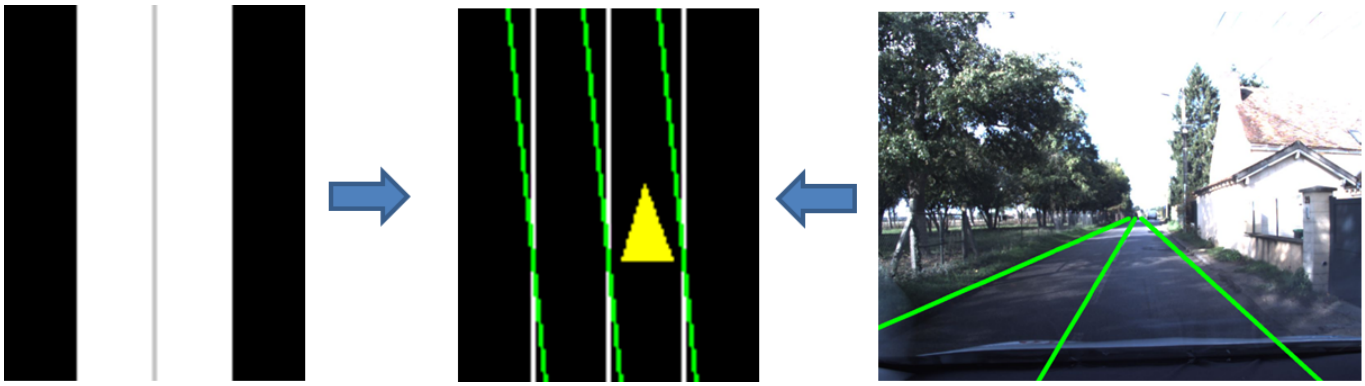


Figure 2.3: Lane level map matching on grid map with lane tracker. a) original map grid, b) lane marking information from image based lane tracker, c) extracted lane markings in white and lane tracker based lane markings projected in green with respect to ego vehicle's pose (yellow triangle).

ICP observation equation is defined in Eq. 2.1.

$$Y_{icp} = \begin{bmatrix} x_{icp} \\ y_{icp} \\ \theta_{icp} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + m_k \tag{2.1}$$

where $x_{icp}$, $y_{icp}$ and $\theta_{icp}$ 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. (2.2).

$$Q_{icp} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & 0 \\ \sigma_{xy}^2 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{2.2}$$

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

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

$\sigma_\theta$ represents standard deviation of vehicle's heading. Detailed derivation of the covariance matrix can be referred to in Najjar et al. [9].

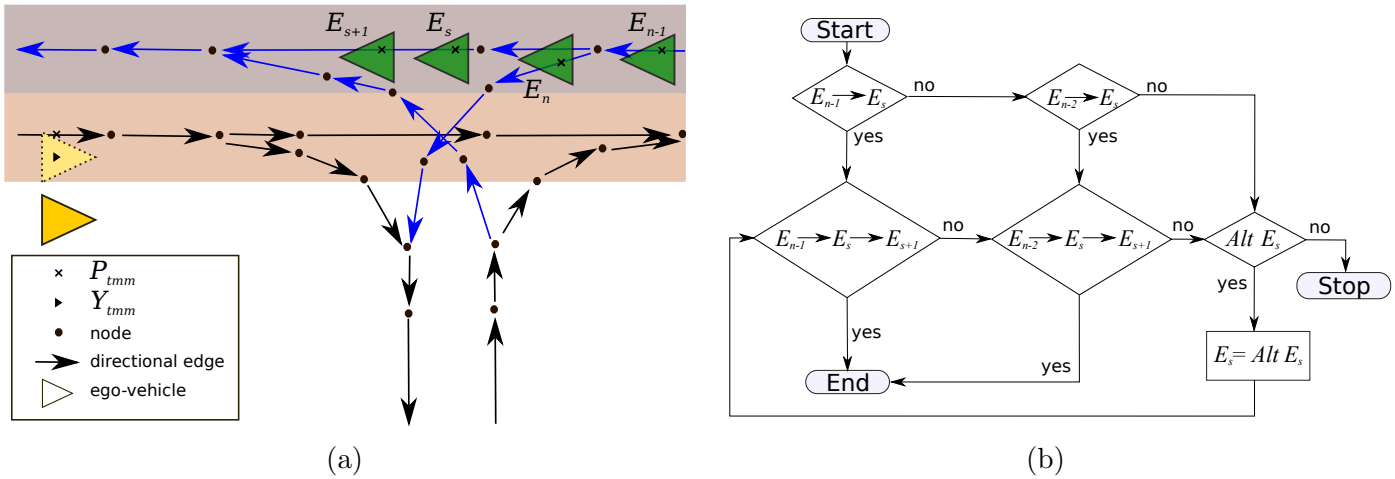
Fault detection scenarios are considered to verify if the ICP corrected pose is suitable for EKF observation. Error metric in ICP is defined by sum of squared distances between corresponding points. If the metric error does not fall below a threshold in a given number of iterations, the ICP is considered 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.

### 2.2.2 Topological map matching

The topological map matching (TMM) algorithm uses if-then decision-rule approach [10] 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. 2.4a. 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.2.4a 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. 2.4a 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. 2.4b. 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




(a) (b)

Figure 2.4: 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.

subsequent edge $E_{s+1}$, this holds true for conflict shown in Fig. 2.4a. 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. 2.2. An example scenario is illustrated in Fig. 2.4a 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. 2.3) is defined in the ENU coordinate frame, similar to ICP observation Eq. 2.1. $n_k$ represents the TMM observation error.

$$Y_{tmm} = \begin{bmatrix} x_{tmm} \\ y_{tmm} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + n_k \tag{2.3}$$

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 [11]. 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

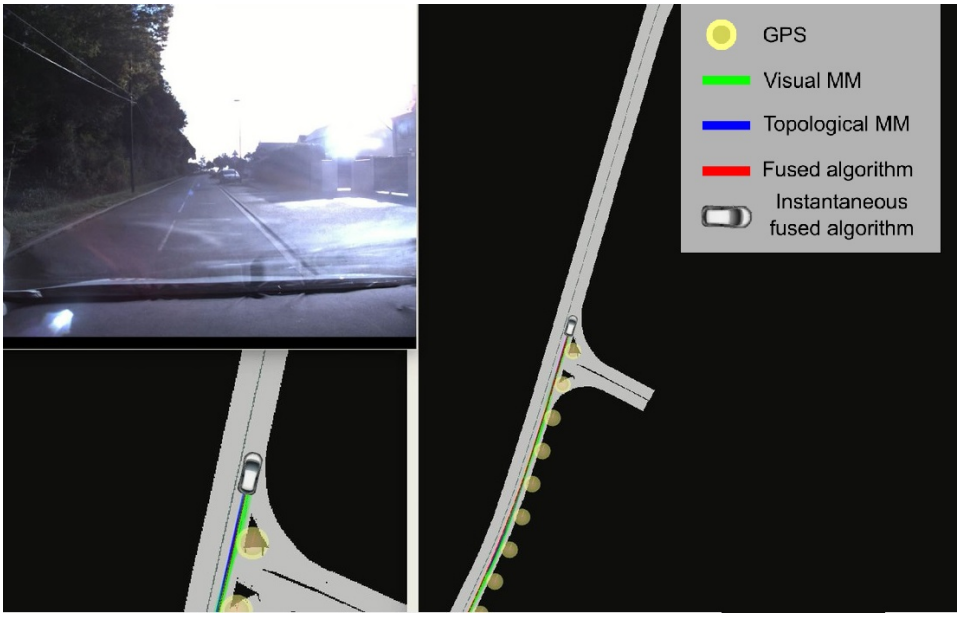



Figure 2.5: Online vehicle localization results on a segment of Route du Bray

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.

The testing and validation of the presented localization approach was also carried out in Rambouillet. Figure 2.5 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.

## 2.3 Percpetion

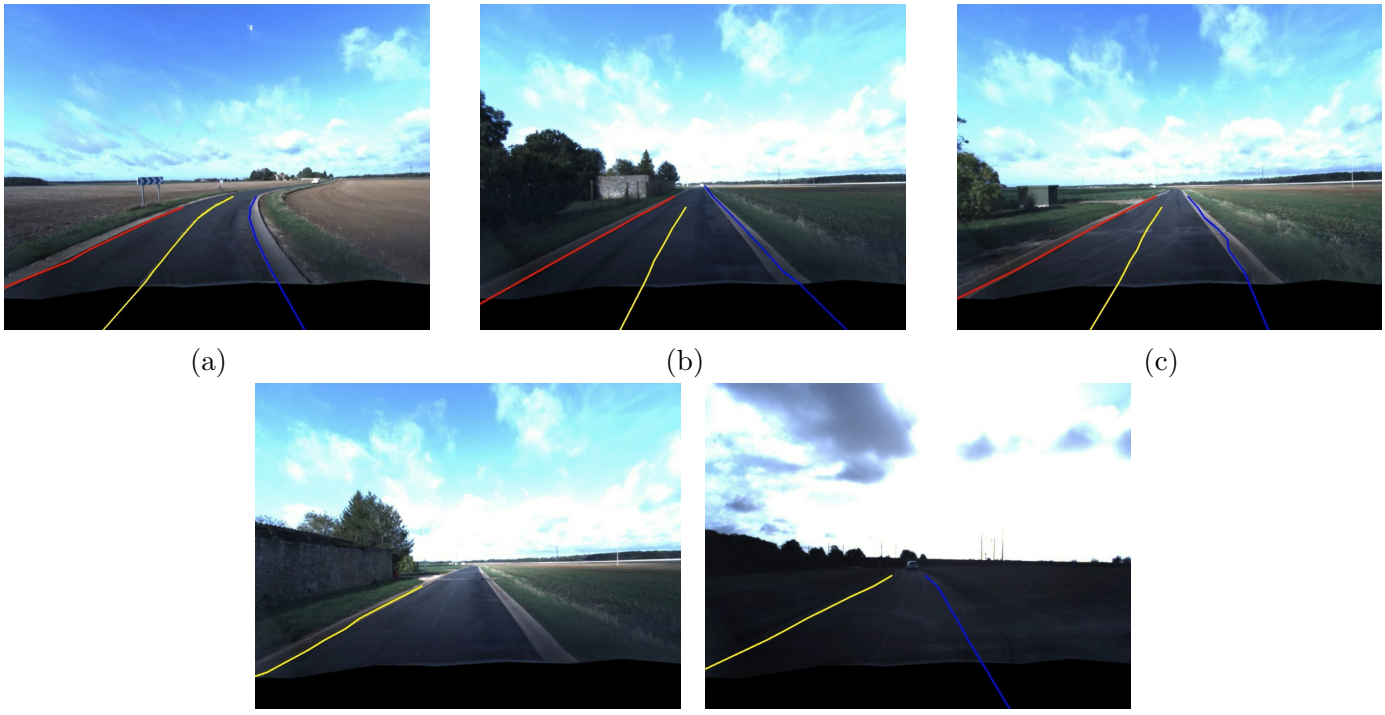
Perceiving the surrounding environment and extracting adequate information is critical for scene understanding and decision-making. The autonomous vehicle must assess the traffic situation and respond to concerns such as: where is the lane and road with respect to vehicle? Is the road curved or straight? 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.2.6a. While NVIDIA DriveWorks delivers great results during varying lighting conditions, it still has limitations. The module does not always recognize road boundaries (Fig.2.6b, 2.6c), a distant lane marking may be mistaken as an ego lane marking (Fig.2.6d), or thin water trail or road repair may be mistaken as a lane marking (Fig.2.6e).

### 2.3.1 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. 2.7.



(a) (b) (c)

(d) (e)

Figure 2.6: NVIDIA DriveWorks lane marking detection.

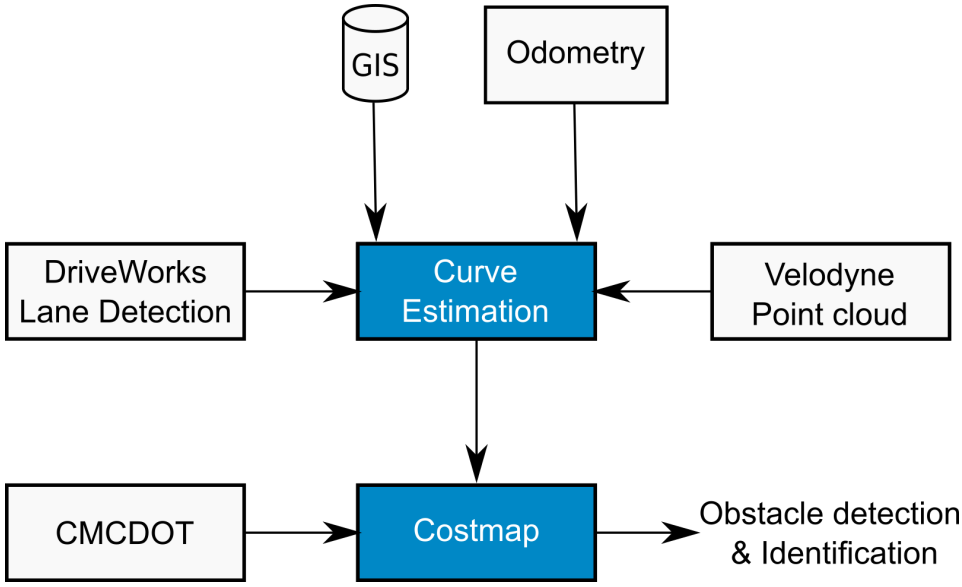
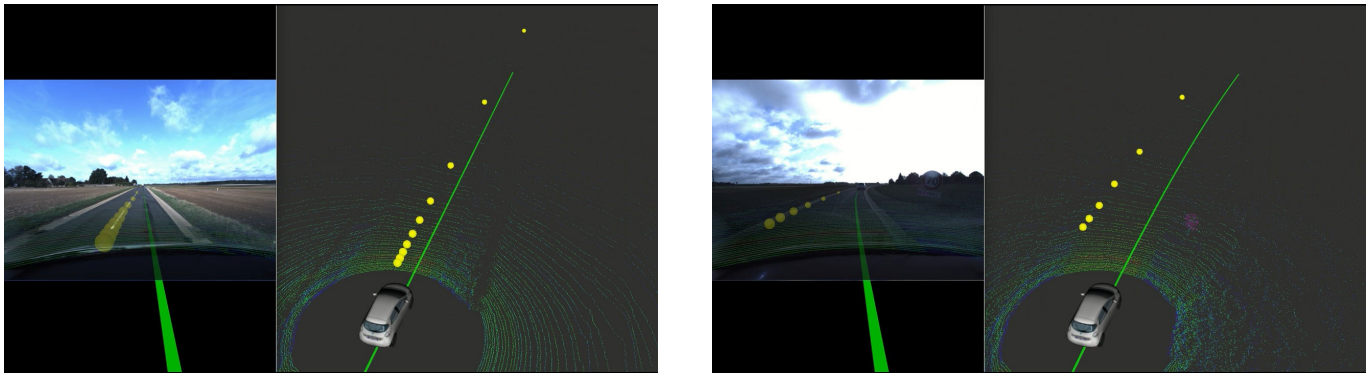



Figure 2.7: Overview of the curve estimation and object detection approach.

Initially, information from NVIDIA 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.2.8 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 2.6b, 2.6c) of Route du Bray, curve estimation is performed based only on the left ego lane markings.



(a) (b)

Figure 2.8: Curve estimation based on NVIDIA DriveWorks and velodyne pointcloud.

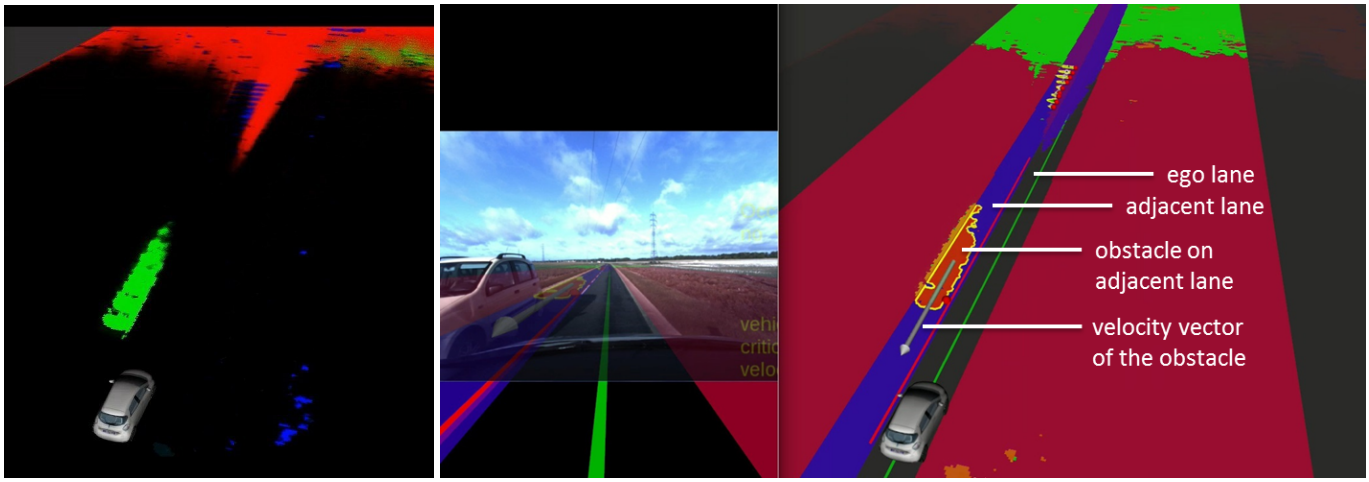
Next, the ego lane makings are filtered before they are fused for road curve estimation. Based on initial or latest curve estimation, the lane marking points are filtered to ensure false ego lane detections such as the ones shown in Figures 2.6d, 2.6e, 2.8b are rejected. If any of the left ego lane makings 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. 2.8, 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.

### 2.3.2 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 [12].

Figure 2.9 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. 2.9b, 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




(a) (b)

Figure 2.9: Costmap generation a) CMCDOT state grid, b) illustrated costmap along with curve estimation.

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.

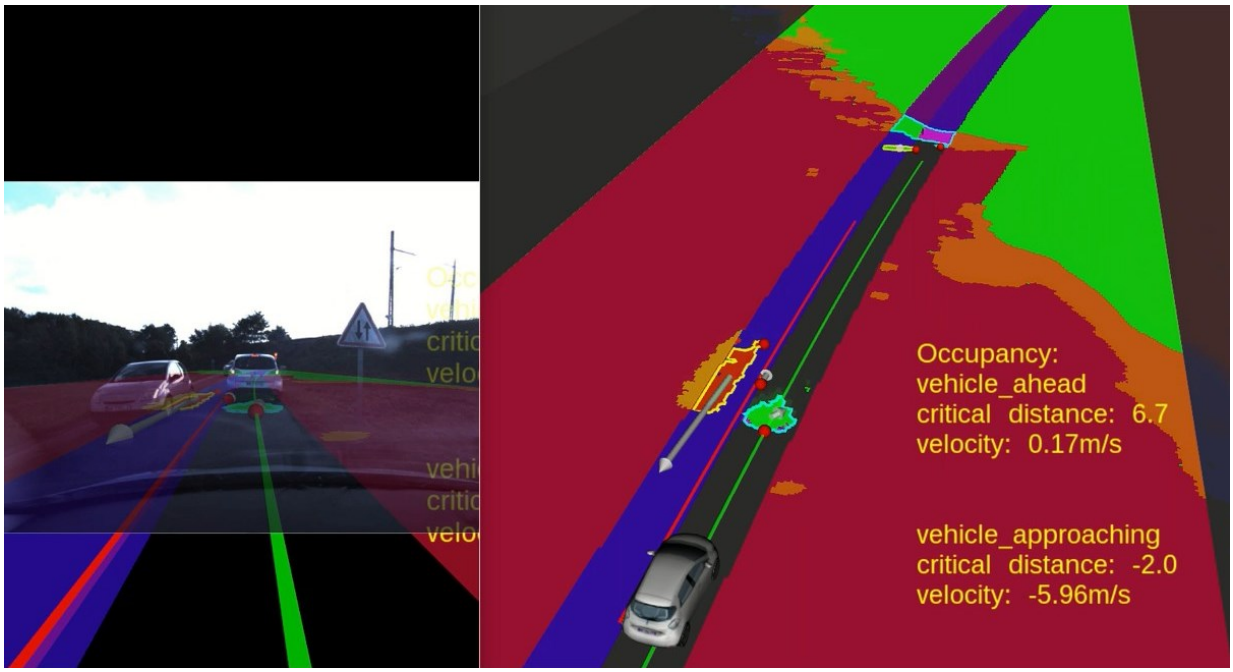



Figure 2.10: Object detection and classification illustrated on the cost map.

Another instant of costmap is shown in Figure 2.10 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 obstacles velocity vector is directed towards ego vehicle. Thus, 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

# Chapter 3

# Hierarchical Decision-Making

Hierarchical decision-making is commonly split intro three tiers:

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

## 3.1 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. 3.1 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.

## 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. 3.2. 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 vehicles position in the next 10 s or next 100 m. Therefore, the planner can transition to the appropriate state such as *Approaching tunnel*.

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.

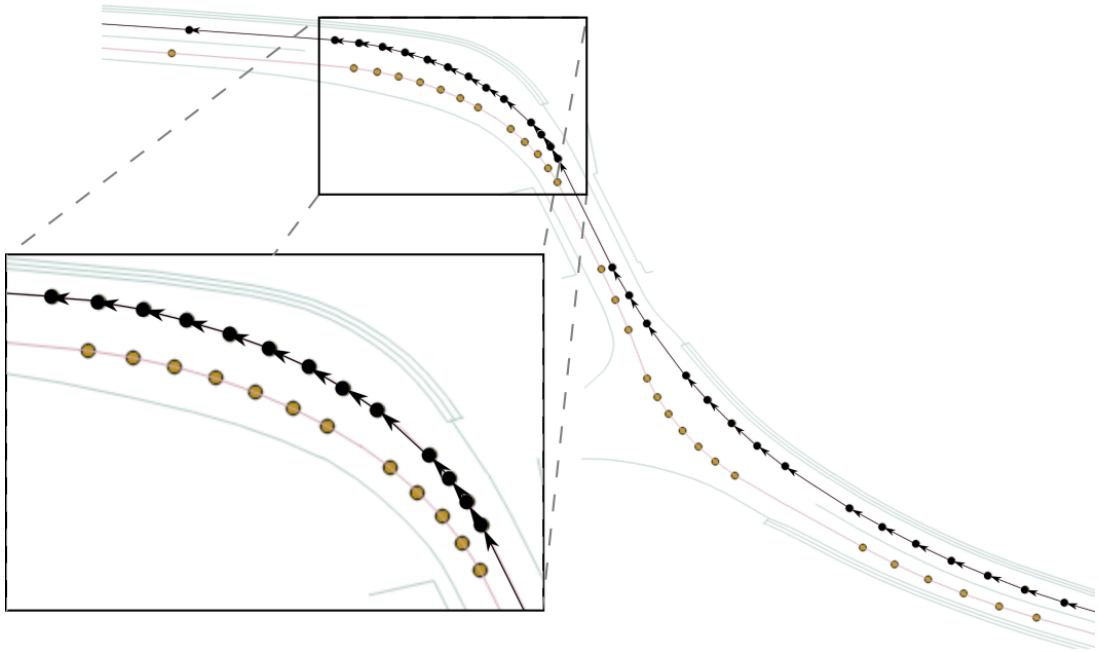


Figure 3.1: Route planning: topological route for crossing the tunnel in direction of Gare de Gazeran.

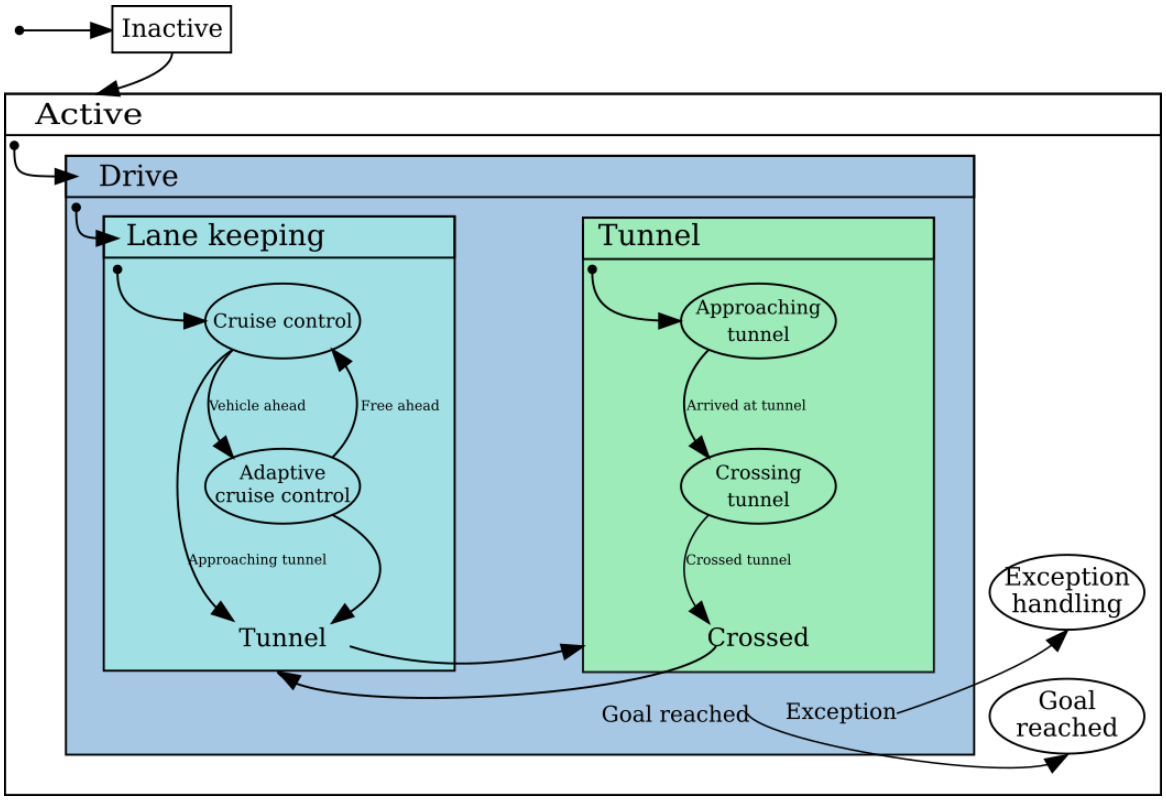



Figure 3.2: Hierarchical finite state machine for autonomous driving on Route du Bray.

## 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:

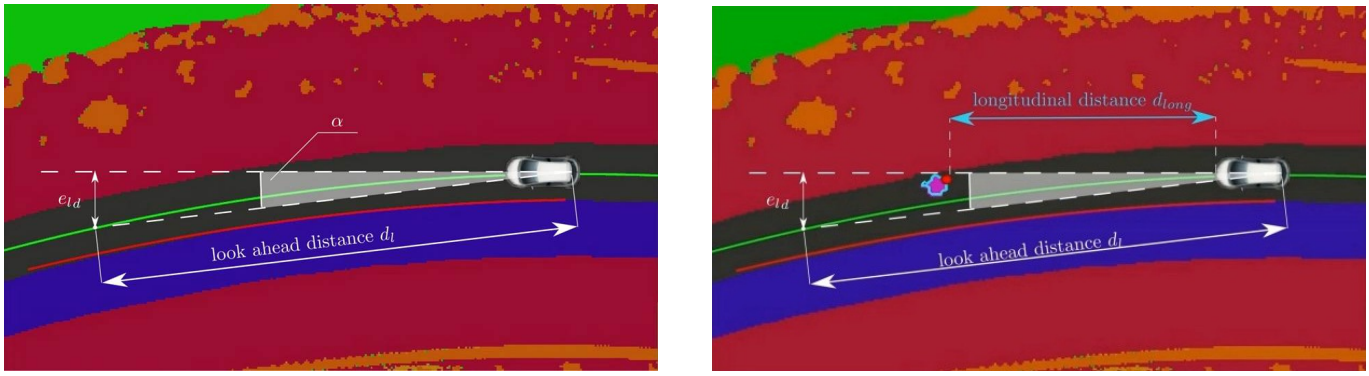
- Lane keeping - cruise control and adaptive cruise control
- Tunnel - approaching and crossing tunnel

### 3.3.1 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. 3.3a 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 3.3: 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 3.4: Transpolis autonomous vehicle testing facility.

The vehicle testing is conducted on the perepherique route marked in green in Fig. 3.4, 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 3.5 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].

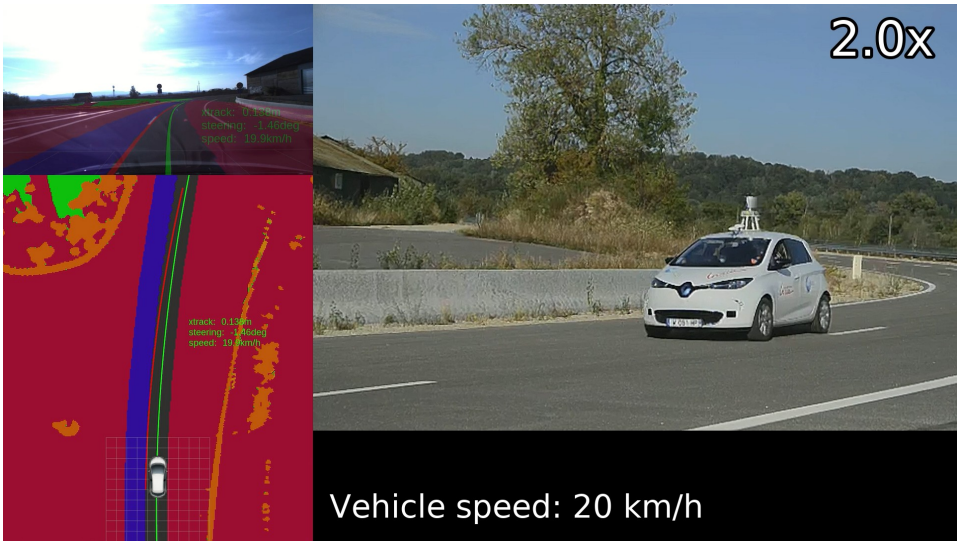



Figure 3.5: Cruise Control testing at Transplis.

### 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 pursuit control, ACC implements longitudinal control to maintain a safe distance to obstacles. Fig. 3.3b illustrates the longitudinal distance, $d_{long}$ to ego lane obstacles.

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

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 3.6 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].

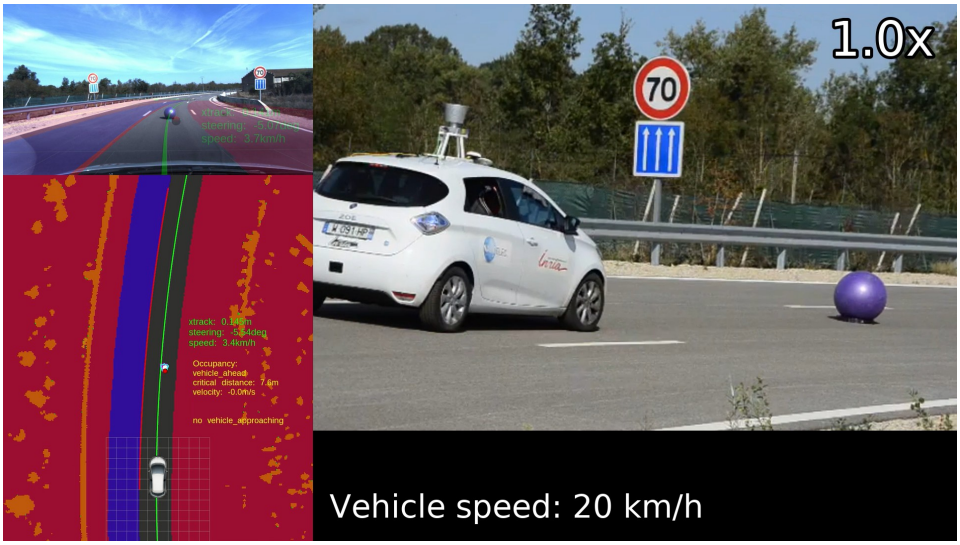



Figure 3.6: Adaptive cruise control testing at Transplis.

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 $\Delta v$. 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 3.7 shows an instant of ACC simulation where Zoe maintains a significant distance of about 20m to the ego lane obstacle. The simulation recordings can be viewed here[2].

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

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

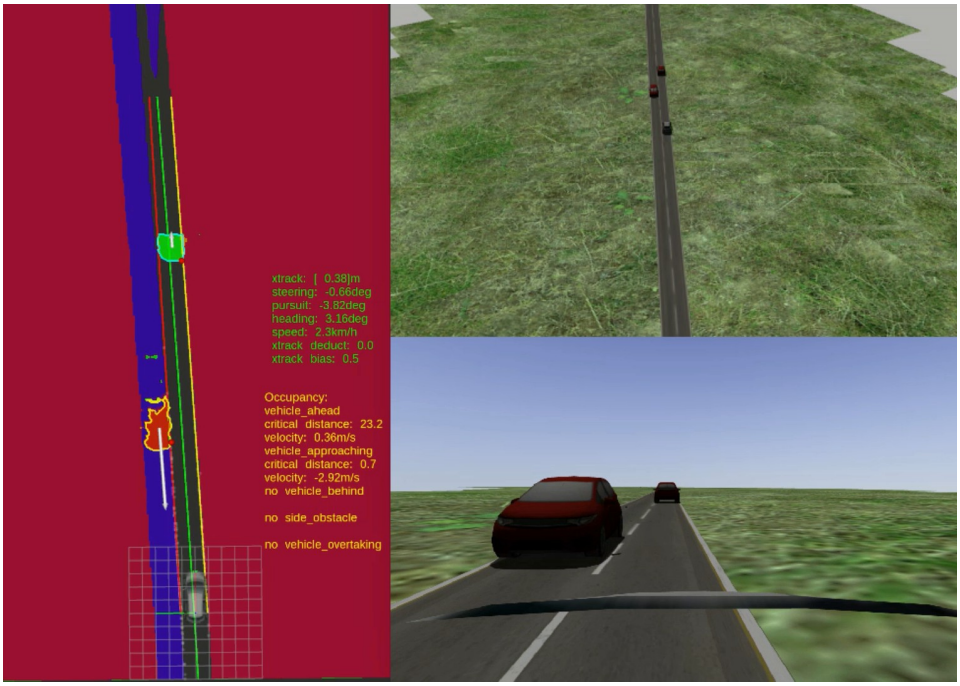



Figure 3.7: Adaptive cruise control with dynamic obstacles in Gazebo simulation.

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.3.7. Once the approaching vehicle(s) has crossed, the ego vehicle returns back to center of lane with trajectory defined by similar parametric curves.

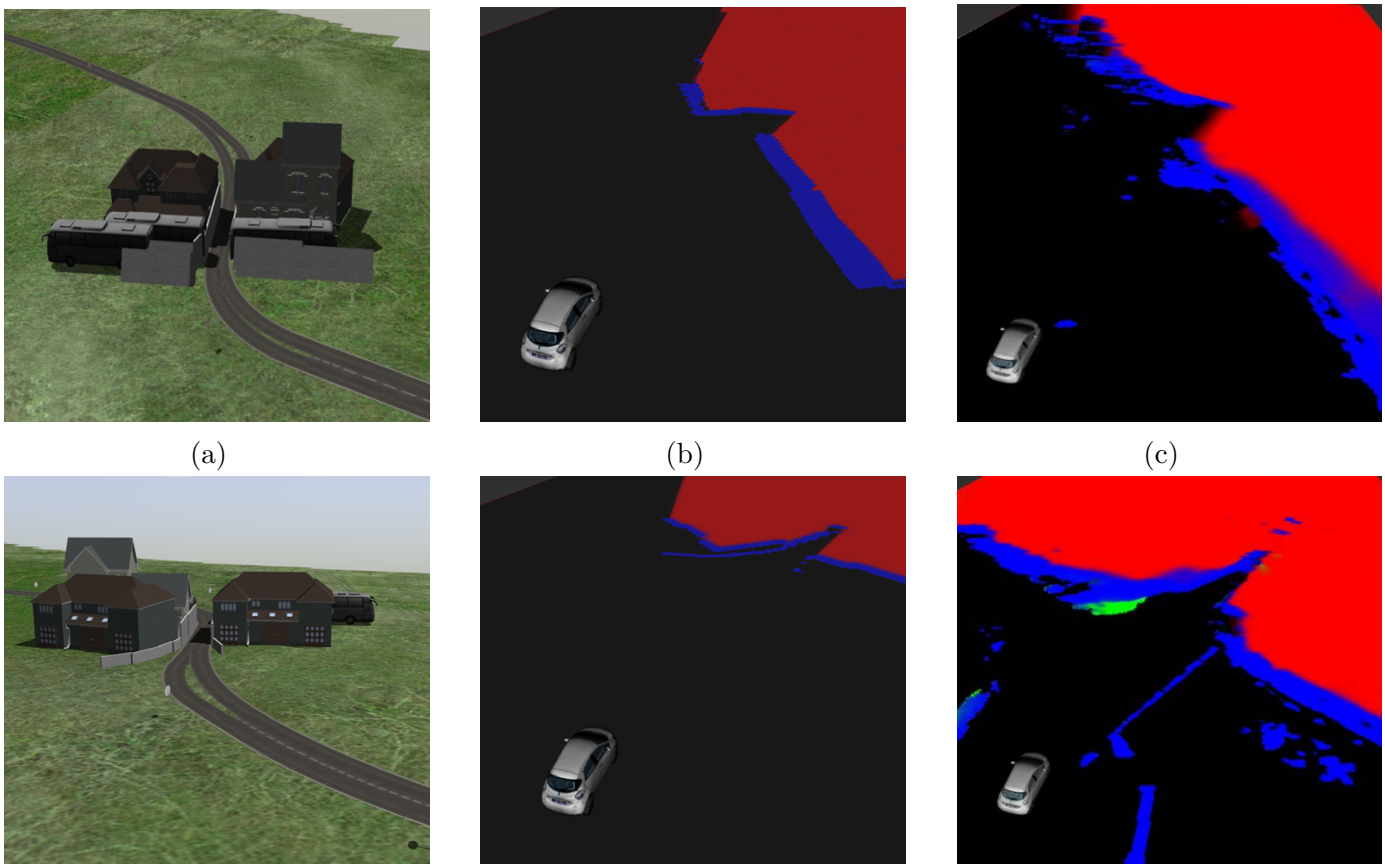
### 3.3.2 Tunnel

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 vehicles 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 3.8: 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. 2.1. The static obstacles are additionally planted to mimic the original tunnel environment. Figure 3.8 shows gazebo environment, CMCDOT state grid generated in simulation and in real condition. Fig 3.8a presents the tunnel setup for direction of travel coming from Gare, with Fig. 3.8b and 3.8c 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. 3.8d, 3.8e, 3.8f.

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 3.9 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. 3.10.

(a) (b)

Figure 3.9: 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).

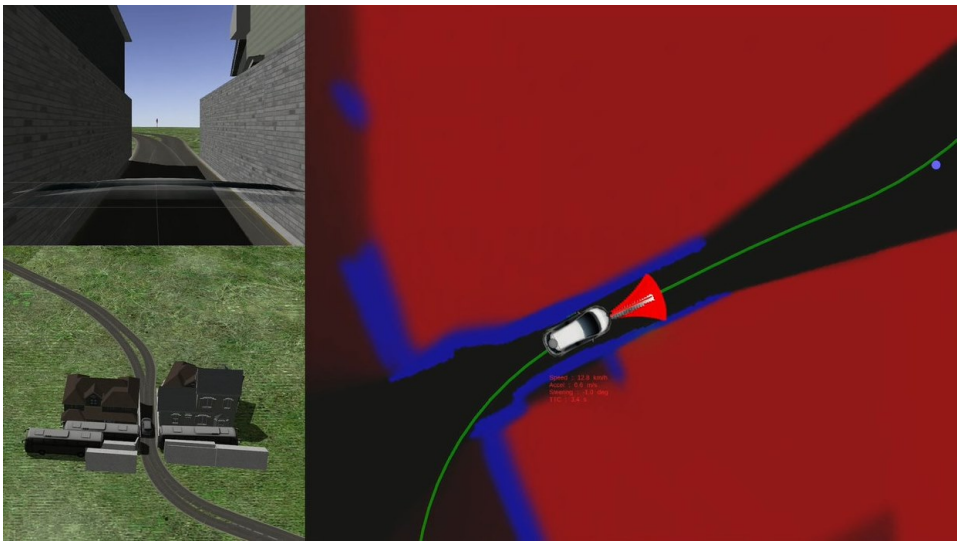


Figure 3.10: Tunnel path corrected and aligned when the vehicle is inside the tunnel.

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 for collision-free trajectories following the target path. Figure 3.11 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].

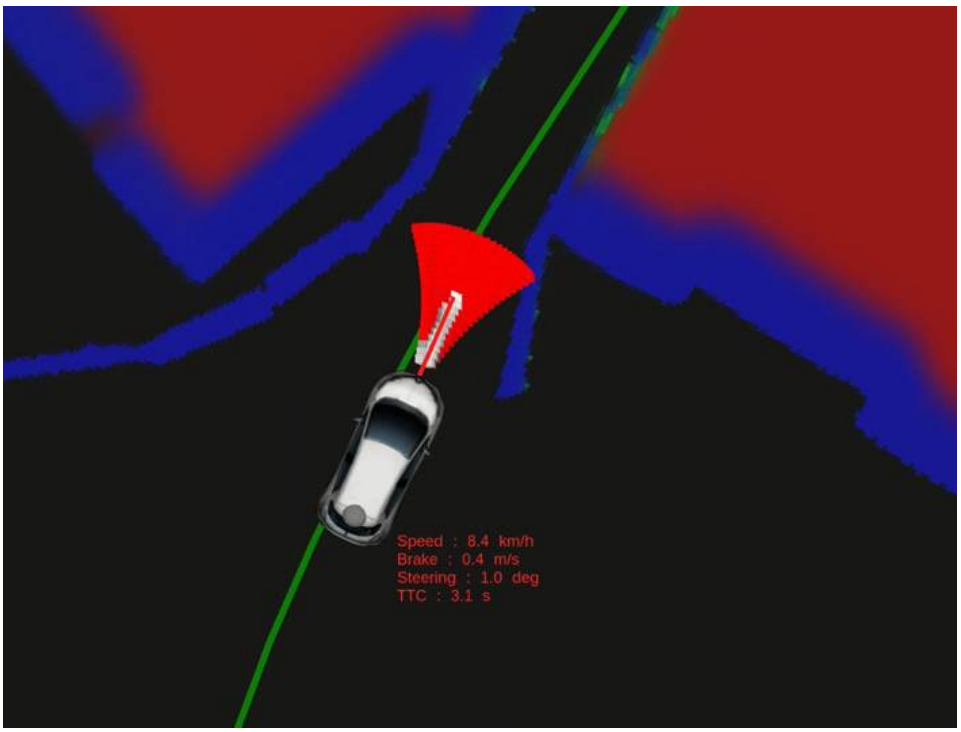



Figure 3.11: Dynamic windo approach planner.

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

# Chapter 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.

## 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 4.1 gives an overview of the mission exchange with the OBU.

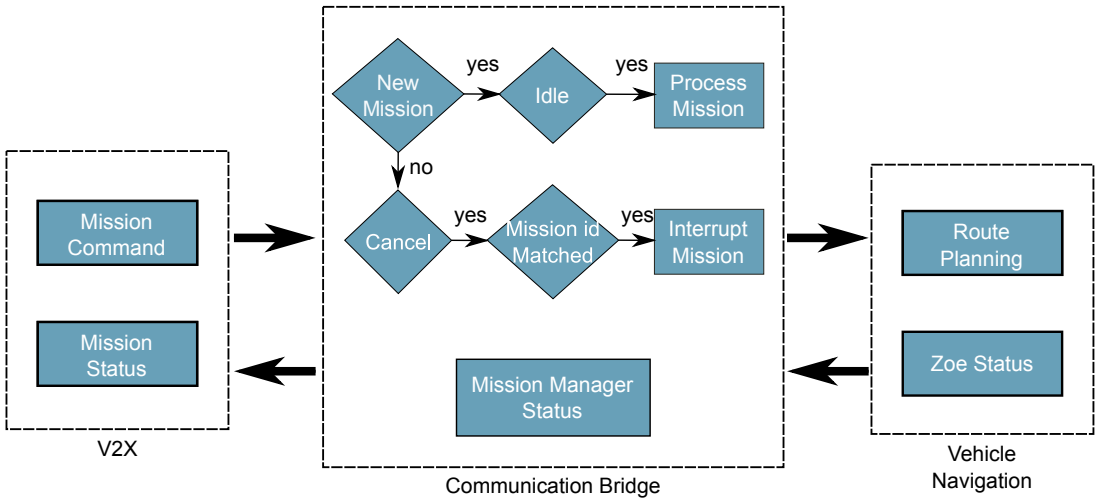



Figure 4.1: V2X communication for mission exchange.

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.

## 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. 4.2. 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.

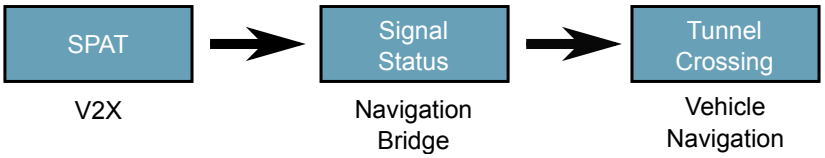



Figure 4.2: V2X communication for traffic signal interface.

# Chapter 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.

# Acknowledgements

This work has been conducted within the ANRT FUI TORNADO project, 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.

Through out the project, Jerome Lussereau has been actively part of all the discussions and organization of resources for the Tornado project. The path planner, developed by Thomas Genevois in team, was integrated in this work to solve the problem of tunnel crossing.

# Bibliography


[1] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 829–846, 2018.

[2] F. Zhang, H. Stähle, G. Chen, C. C. C. Simon, C. Buckl, and A. Knoll, "A sensor fusion approach for localization with cumulative error elimination," in *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2012, pp. 1–6.

[3] M. Hashemi and H. A. Karimi, "A critical review of real-time map-matching algorithms: Current issues and future directions," *Computers, Environment and Urban Systems*, vol. 48, pp. 153–165, 2014.

[4] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments." in *Robotics: Science and Systems*, vol. 4. Citeseer, 2007, p. 1.

[5] A. Mueller, M. Himmelsbach, T. Luettel, F. v. Hundelshausen, and H.-J. Wuensche, "Gis-based topological robot localization through lidar crossroad detection," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 2001–2008.

[6] T. Weiss, N. Kaempchen, and K. Dietmayer, "Precise ego-localization in urban areas using laser-scanner and high accuracy feature maps," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* IEEE, 2005, pp. 284–289.

[7] 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.

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

[9] 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.

[10] 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.

[11] R. Asghar, M. Garzn, 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.

[12] 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.