# Maps and their impact on the functional safety of automated driving

Pai, V.N.

*Award date:*
2021



Department of Mathematics and Computer Science
Model Driven Software Engineering Research Group

# Maps and their impact on the functional safety of automated driving

*Master Thesis*

Vishwanath Nagnath Pai

Supervisors:
dr.ir. Ion Barosan, TU/e
Prof.dr. M.G.M Van den Brand, TU/e
dr.ir. LMT (Bart) Somers, TU/e
dr.ir. Arash Khabbaz Saberi, TomTom
Emrah Eminoglu, TomTom

Eindhoven, August 2021

# Abstract

There is a wave of advancement approaching the automotive industry, with the impetus shifting from manually-driven to automated driving systems. The role of the driver is steadily being decreased by incorporating automated driving systems in the vehicle. The levels of automation have brought about ADAS systems such as adaptive cruise control and lane assists. These complex systems could utilize maps as a sensor to make real-time decisions. OEMs will need to rely on the quality of maps and sensor data to ensure their systems are safe and reliable in different operating conditions.

The thesis is aimed at observing the impact of the map on the functional safety of automated driving. A safety analysis, System Theoretic Process Analysis (STPA), is conducted on a SAE Level 2/3 automated driving vehicle using maps. The objective of the analysis is to estimate the different unsafe scenarios caused due to map data. The identified list of scenarios is validated using TomTom's data sources. The list of scenarios is further used to identify safety-critical map features, which would serve as the focus of the simulation. Uncertainties in the map, by injecting a Gaussian noise signal, are simulated using an autonomous driving simulator, CARLA. The safety of the vehicle is evaluated by setting key performance indicators and recording their respective values in different test cases.

The results of this thesis would aid in the identification of safety-critical features of the map. The proposed methodology can be reapplied to vehicles possessing higher levels of automation. Furthermore, conclusions drawn from the simulation would emphasize the requirement of safety and quality management systems for maps.

# Preface

During this thesis, I have had the opportunity to work with a lot of different people. These people have provided me with a great deal of support and assistance.

I would first like to thank my supervisor, Dr.ir. Ion Barosan, whose expertise was invaluable in formulating the research questions and methodology. I would like to thank you for being a constant source of motivation during this thesis. Your insightful feedback pushed me to sharpen my thinking and pushed me to work at a higher level.

I would like to acknowledge my supervisors at TomTom, Dr.ir. Arash Khabbaz Saberi and Emrah Eminoglu for their wonderful collaboration. I would like to thank them for their patient support and all the opportunities I was given to perform my research. I would like to thank them for giving me the freedom and flexibility to build my skill-set and utilise my creativity in the course of this project. Special thanks go out to the members of the Safety Team, who provided me support and invaluable feedback during my thesis.

I would like to thank my committee members, Prof.Dr. M.G.J. van den Brand, and Dr.ir. LMT Somers for reviewing and evaluating my thesis project. I am grateful and I express my utmost gratitude towards them.

Last but not least, I would like to thank my family, who have been miles away, for their tremendous support and constant motivation during my thesis. I would like to thank Deepa Hegde, for her faith and undying support during this challenging journey. I would also like to thank my peers who have helped me in pushing through the difficult parts of my project

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this chapter, a brief introduction to the stakeholders responsible for this project, i.e., TomTom, is provided. TomTom's map production system is presented following the introduction. The results of the background information concerning different safety analysis techniques are presented. This is followed by describing the given problem statement. Research questions are identified from the given problem definition and are presented in this section. In closing, the outline of the report is discussed.

## 1.1 Introduction to TomTom

TomTom is the leading location technology provider and is shaping mobility with highly accurate maps, navigation software, real-time traffic information, and services [1]. TomTom is well recognized for developing a variety of navigation products such as Advanced driver-assist system (ADAS) and High definition (HD) maps [2]. In addition to HD and ADAS Maps, TomTom also provides point cloud maps that could be used by others for third-party map generation [1].

## 1.2 Map production system at TomTom

TomTom develops an array of navigation products such as ADAS and HD maps. In addition to HD and ADAS Maps, TomTom also provides point cloud maps that could be used by others for third-party map generation. Gaming and driving automation simulations could be run by using the products created by TomTom. ADAS maps are utilized for driving various driver assist systems such as adaptive cruise control, whereas HD maps are designed specifically for a SAE level 3 autonomous vehicle [3]. The map consists of a wide variety of features such as road gradients, lane information, traffic signs, speed limits which would aid in driving a vehicle of such level of automation [4]. To produce the map, a system has been established which is responsible for data sourcing, map production, and delivery. The output of the system yields a map that can be delivered to the customers and OEMs. The architecture of the HD map production system is illustrated in Figure 1.1.

Observational data from roads is gathered from various sources. TomTom utilizes its self-developed sourcing technology with the help of the Mobile Mapping (MoMa) System [5]. Vehicles equipped with multiple sensors such as LiDAR, cameras, radar, IMU are driven across various roads. Data gathered from these vehicles is assessed by conducting quality checks. Once it is approved, the data is ingested through the ingestion process [6]. The assessed data is uploaded to the TomTom Cloud. This is followed by a real-time assessment of the uploaded data. The processed data is now supplied to the map production system.

The map production system is responsible for extracting all the appropriate features from the processed data and update the current map repository, which has already been established [3]. Initially, stationary features such as traffic lights, speed limits are identified. This is followed by

the classification of these features based on their type, i.e, grouping all the speed limit signs. The methods used for grouping features vary with the type of observation [3]. The grouped features are then quantified. The quantification of features aids in their storage in the observation archive. Grouped observations present in the archive are selected based on time, location, and confidence, accuracy levels. The selected observations are fused to yield observations with higher levels of accuracy and confidence. Reality changes can be detected after receiving a single high confidence observation or many less confident observations of the same changed reality. If the changes are not detected, then either reality hasn't changed, or insufficient checks were performed. Once this step has been completed, the map is compiled according to the defined rules related to the data model and capturing. The initial map is developed, and updates are applied over it based on any detected reality changes.

The maps are delivered to customers and OEMs through two main sources, AutoStream and NDS Cloud delivery. AutoStream is a map streaming method in which the maps are streamed along the route of the car [4]. The software can stream a varying layer of maps (HD/ADAS) depending on the functionality available on the car.

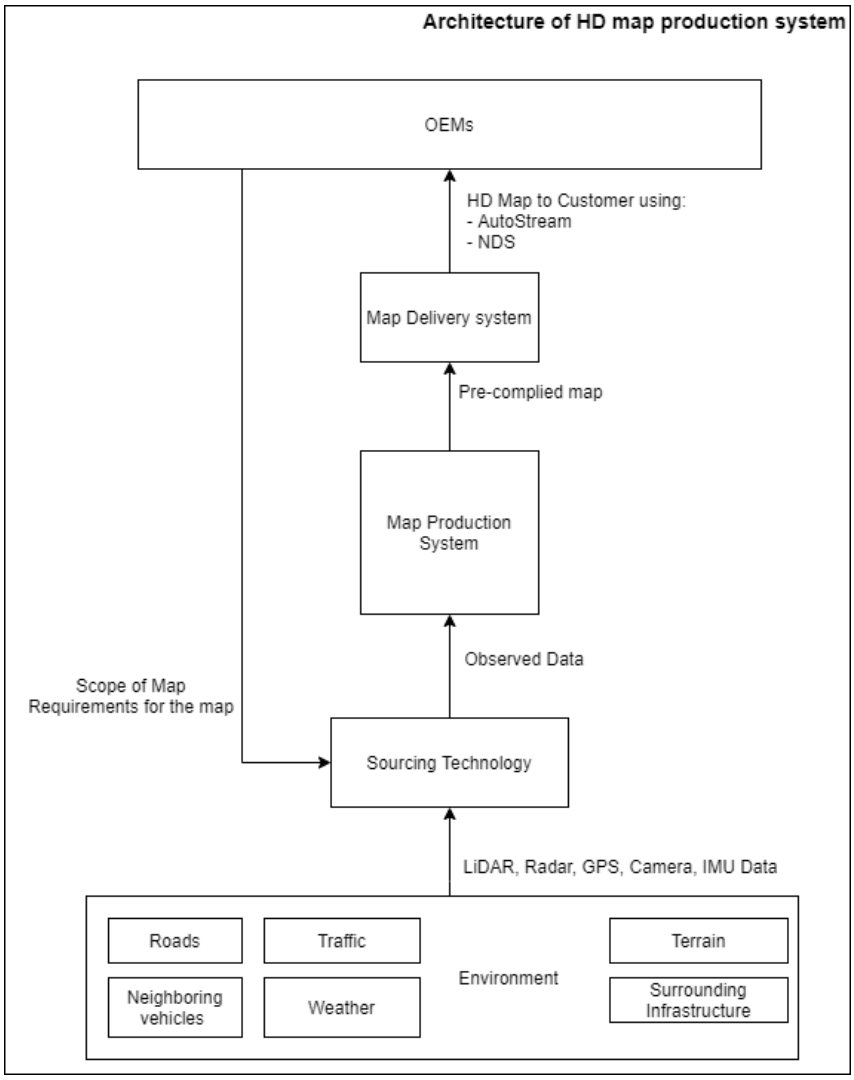


Figure 1.1: Architecture of TomTom's High Definition (HD) Map production system.

## 1.3 Methods of safety analysis

Automated driving vehicles present a tremendous challenge in ensuring the reliability in performance and safety for not just the passengers but the surrounding vehicles. These modern vehicles consist of complex mechanical systems, which work in unison with other software-driven systems [7]. However, the performance of such vehicles depends heavily on the availability of maps [8]. Automated driving vehicles utilise various features of the map, such as the lane features. Advanced Driver assist systems (ADAS) such as adaptive cruise control use map features about "speed limit" data to ensure the vehicle not only maintains a specified distance concerning the vehicle ahead, but also a set speed. Automation of the driving process is achieved by depending on a fusion of map and vehicle sensor data. Thus, the safety of the automated driving systems and maps being provided must be ensured. Safety of systems are analysed using different types of safety analysis techniques.

They can be categorised based on the flow of analysis, which are as follows:

1. **Top-down approach**

   In this approach, the goal of the analysis is to identify the failure modes of a system at the highest level. This is followed by a low-level abstraction of the system, which is used to identify the remaining failure modes of the system occurring at lower levels in the system. Examples of top-down approaches are System Theoretic Process Analysis (STPA), and Fault Tree Analysis (FTA).

   (a) *STPA* is a safety analysis method based on extended model of accident causation. STPA works under the assumption that accidents can occur not only due to system failures, but due to unsafe interactions between different components of the system [9]. This can be applied in the early design stages of a system. The system continues to perform its functions as per specification, however a hazard is still encountered. It includes software and human operators in the analysis, thus ensuring that all possible causal factors for the occurrence of losses are included [9].

   (b) *FTA* is a hazard identification tool. It utilises a top-down approach. It is used to solve a wide range of problems. FTA is performed by constantly asking the question: how can a specific hazard occur, and what are the potential causes of this event [10]. FTA is a graphical logical rendition of fault events that may occur in a given system [11].

2. **Bottom-up approach**

   In this approach, the goal of the analysis is to identify the failure modes of a system at the lowest level. This process is extended to higher levels in the system. An example of bottom-up approach is Failure mode effect analysis (FMEA).

   *FMEA* is a bottom-up reliability analysis method. The analysis starts from unit-level and moves upwards towards a system-level view of the system [12]. FMEA relies on brainstorming to identify different failure modes and their effects on higher levels of the system [13]. There are several types of FMEA which can be performed such as process FMEA, design FMEA, and system FMEA. The goal of FMEA is to identify failures at a unit level of the system which could lead to vehicle-level hazards. The failures detected are used for deriving safety requirements.

Conventional safety analysis techniques such as Failure mode effect analysis (FMEA) and System Hazard analysis (SHA) are applied particularly on mechanical or hardware-driven systems [14]. Furthermore, they rely on probability theory, to identify the effects of each components' malfunction on the remaining components [15]. As the level of technology in vehicles advance, the components of the vehicle driven using software-driven components will also increase [16]. There will be multiple communication protocols in place to ensure smooth communication between different systems. Unsafe interaction between these software-driven components can occur, which

would result in unsafe scenarios for the vehicle. These scenarios would remain undetected if conventional analysis techniques were applied to these systems. Figure 1.2 presents the commonality between both the approaches presented above with respect to the identified failure modes. Therefore, performing a top-down approach technique such as STPA would be a useful addition to the existing safety analysis techniques.

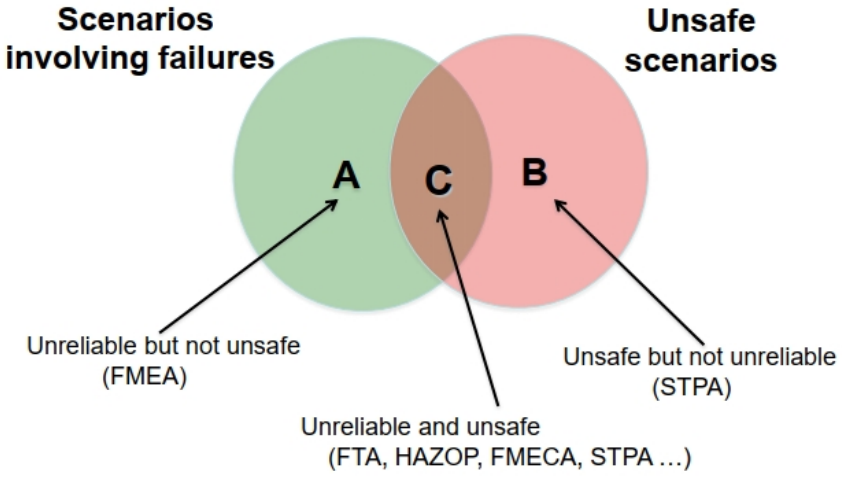


Figure 1.2: Identification of failure modes using STPA and FMEA.

## 1.4 Problem definition

In the automotive sector, conventional safety analysis techniques such as FMEA are applied to developed systems. The results of the analysis are implemented within these systems to ensure safety against failures occurring in the lower levels of the system [13]. An automated driving vehicle comprises multiple complex software-driven systems, which communicate with each other [7]. Unsafe scenarios could occur in these channels, which would go unnoticed in a conventional safety analysis. As illustrated in Figure 1.2, FMEA does not yield scenarios occurring due to unsafe interaction between systems. Therefore, an improved method of safety analysis must be performed to ensure the all-around safety of the system. This would make sure the system implemented performs optimally in a variety of scenarios, thereby keeping the driver and the passengers safe.

This project aims to study the HD map production system developed by TomTom and how can maps lead to unsafe scenarios in automated driving system. The level of automation under consideration is Level 2/3 as per the defined SAE standards for automation in vehicles [17]. The study is performed from the point of view of map-makers. The map production system was studied to develop an understanding of the products created by TomTom. The method of safety analysis selected is System Theoretic Process analysis (STPA). The research can be broken down into two main elements, the safety analysis and the simulation. A safety analysis is conducted using the selected technique, STPA, of a given SAE Level 2/3 automated driving vehicle, which uses map in the automation process [17]. This is followed by performing simulations in an autonomous driving simulator, to observe the impact of map uncertainty on the functional safety of the vehicle. The given problem statement has been broken down into two research questions. The research questions are as follows:

1. *In what scenarios does the traffic and lane features of the map impact the functional safety of automated vehicles?*

2. *In the event of camera failure, what is the dependency of lateral control of an automated driving vehicle on the quality of accurate maps?*

## 1.5 Thesis outline

This report is split into four parts. Chapter 2 consists of the background information, which gives a detailed description about the selected safety analysis technique, STPA, and the choice of autonomous driving simulator. Chapter 3 consists of the methodology, which presents the application of STPA and autonomous driving simulator to the given problem statement. In chapter 4, the results of the safety analysis are presented. This chapter includes the validation of the results of the safety analysis. Chapter 5 presents the results of the simulations of map uncertainty conducted in CARLA. Key remarks and observations drawn from the chapters 4 and 5 are presented in Chapter 6, which is titled discussions. Furthermore, the limitations of the project are highlighted in this chapter. Chapter 7, which is titled Conclusions and future works, is used to describe the conclusions drawn from this thesis. This is followed by discussing the future works of the project.

# Chapter 2

# Background Information

In this chapter, a detailed description about the selected safety analysis technique, which is System Theoretic Process analysis (STPA), is given. This is followed by a discussion of different self-driving simulators available, which is used in the selection of a simulator for the given application. The various features of the selected simulation environment, which are used in this study, are presented.

## 2.1 System Theoretic Process Analysis

System-Theoretic Process Analysis (STPA) is a hazard analysis technique based on an extended model of accident causation [9]. In addition to the failure of components, STPA works under the assumption that accidents can be caused by unsafe interactions of system components, none of which may have failed. In the STPA framework, a system will not enter a hazardous state unless an unsafe control action has been performed by the controller. The steps performed in the STPA approach have been represented in Figure 2.1. Each of the steps will be explained in further detail in the upcoming sections.

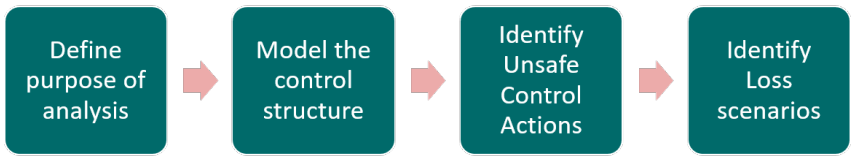


Figure 2.1: Steps undertaken in STPA.

### 2.1.1 Define purpose of analysis

STPA is performed on a system for a given purpose. The purpose of this approach must be defined with clarity, which would aid in yielding appropriate results. The scope of the approach must be defined along with its purpose. The scope of the analysis pertains to the system which is under investigation. System definition consists of identifying the elements of the system which will be analysed through this approach. The definition of the system must include the system boundary and its interactions with other systems and the environment. The system boundaries are identified by considering parts of the system over which the system designer has some control [9]. The system can be visualised using a hierarchical control structure which would capture the feedback

control loops [13]. The steps followed while defining the purpose of the analysis are highlighted in Figure 2.2.

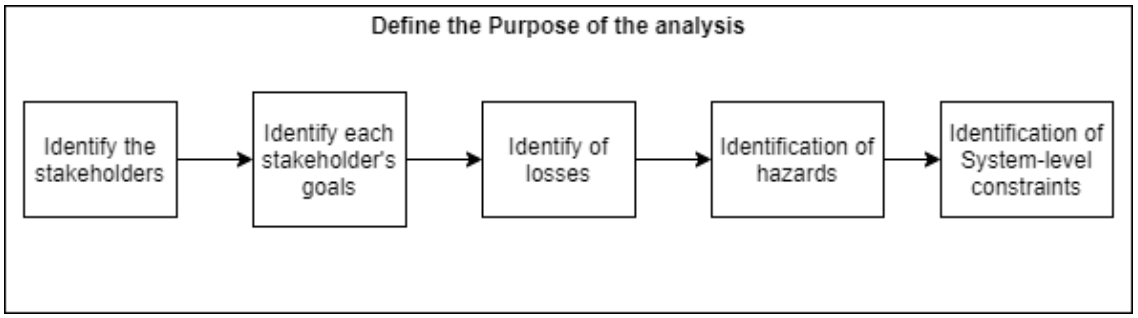


Figure 2.2: First step of STPA.

**Identification of Losses:** A loss involves something of value to the stakeholders [9]. Losses may include a loss of human life or human injury, property damage, environmental pollution, or any other loss that is unacceptable to the stakeholders [9]. The goal of this approach is to prevent losses. The analysis begins by identifying all the involved stakeholders. This is followed by defining goals for each stakeholder, which is then translated into a loss. If multiple losses are present for each stakeholder, they can be ranked based on their importance [15]. STPA can be performed specifically to prevent losses which are objectionable. The next step is to define hazards related to these losses.

**Identification of hazards:** A hazard is a system state or set of conditions that, together with a set of worst-case environmental conditions, will lead to a loss [9]. Using the defined system and its boundaries, system-level hazards can be listed by identifying system states or conditions that will lead to a loss in the worst case environmental condition [18]. A hazard can be traced to multiple losses. Criteria considered whilst defining hazards are as follows:

1. Hazards are system states or conditions.
2. Hazard must describe the state or conditions to be prevented at the system-level and not sub-system or component level.
3. Hazards will lead to a loss in some worst case environmental condition.

**Identification of System-level constraints:** A system-level constraint specifies system conditions or behaviors that need to be satisfied to prevent hazards [9]. The defined system-level hazards are inverted to define their respective constraints. Each constraint can be traced to multiple hazards. As stated previously, hazards can also be traced to multiple losses. Therefore, a constraint can directly prevent one or more losses. Furthermore, constraints must be not defined in the form of a particular solution. Specifying of a solution during constraint definition could lead to better solutions being overlooked.

### 2.1.2 Model Control structures

The second step of the analysis consists of modelling control structures which will be used for identifying control actions, unsafe control actions and the controller constraints. A hierarchical control structure is a system model that is composed of feedback control loops [9]. A hierarchical control structure is composed of control loops. A control loop is formed when a controller provides control actions to control some process and to enforce constraints on the behaviour of the controlled process. The types of elements considered while modelling control structures are as follows [9]:

1. Controllers

2. Controlled processes
3. Control actions
4. Feedback
5. Other inputs and outputs from components within the system

The control structure must be modelled with a defined set of rules. The vertical axis of the control structure is used for indicating control and authority within the system. The element at the top of the control structure possesses the highest level of authority. Each element has the authority to control elements below it. Downward arrows from system represent control actions and upward arrows represent feedback.

The modelling of control structure begins with a high-level abstraction of the system under analysis, thereby aligning with the top-down approach followed in STPA. High-level abstraction of the system refers to the modelling of the system and its interactions with other systems and the environment. The high-level abstraction is followed up by modelling control structures at the subsystem or lower level. With each iteration of the control structure, more details are added. This leads to a finer and accurate representation of the system and its components. The steps performed following the modelling of control structures are presented in Figure 2.3.

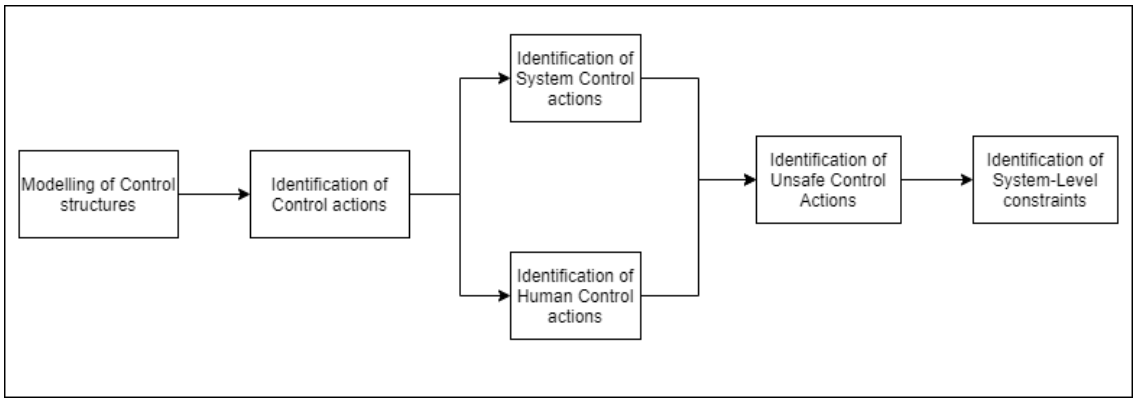


Figure 2.3: Steps II and III of STPA.

**Identification of control actions:** Modelled control structures are used for defining the functions performed by each entity in the structure. These functions are referred to as control actions. Control actions can be split into the following categories:

1. System control actions are control actions that are performed by elements in the system [9]. An example of a system control action would be an action performed by a controller to control a given process.
2. Human control actions are control actions that are performed by human actors in the system [9]. An example of a human control action would be an action which is performed by a human actor present in the system.

Control actions are listed out by describing the function performed by the element in the system and the communication of its resultant output with the other elements in the system. A controller can make decisions using its defined control algorithm and feedback from the controlled process. Thus it is vital to include feedback identification during the listing of control actions.

### 2.1.3 Identification of Unsafe Control Actions (UCA)

The listing of control actions is followed by the identification of unsafe control actions. An Unsafe Control Action (UCA) is a control action, that in a particular context and worst case environment,

will lead to a hazard [9]. There are four ways in which a control action can be termed as unsafe [9][15]:

1. **Not providing the control action leads to a hazard:**

   This consists of cases wherein the system or elements of the system do not provide a required control action.

2. **Providing the control action leads to a hazard:**

   This consists of cases wherein the system or elements of the system provide a control action, but it still leads to a hazard. The following cases could be considered in this step:

   (a) The control action may not be safe given the situation the system is in.

   (b) The control action performed by the system utilises an incorrect input or provides an incorrect output.

   (c) The control action performed by the system may be insufficient or excessive.

   (d) The control action performed by the system may result in an output in the opposite or unsafe direction.

3. **Providing a potentially safe control action but too early, too late, or in the wrong order:**

   This consists of cases wherein the system performs a control action outside of its required timing bounds. The system may perform a function before the scenario warrants such action from it.

4. **The control action lasts too long or is stopped too soon (for continuous control actions, not discrete ones).**

   This consists of cases wherein the system performs a control action for too long or too short a period before the system reaches the required state, thus endangering the system. The system may provide the required output for a duration longer or shorter than expected, thereby putting the system at risk of causing a hazard.

Unsafe control actions are identified along with the respective hazards they would be causing. This provides traceability of hazards with the unsafe control actions. Hazards can be considered as the backbone of the analysis since all the steps can be traced back to them.

**Identification of controller constraints:**
The list of unsafe control actions can be translated into controller constraints. This process is similar to what has been described in Section 2.1.1. Unsafe control actions are inverted to yield constraints for each system which would perform them given the worst case environment. By listing out constraints, the controller would be prevented from performing unsafe control actions which would compromise the overall safety of the system.

### 2.1.4 Identification of loss scenarios

Loss scenarios describe the situation that can lead to the unsafe control actions to and to hazards [9]. The two types of scenarios which are identified in this step are:

1. **Why would unsafe control actions occur?**

   Scenarios in this category can be identified by selecting an unsafe control action and moving backwards to understand what could be the reason behind the controller providing or not providing the control action. Unsafe control actions could also be provided due to unsafe behaviour exhibited by the controller. There are four reasons which can be traced to unsafe controller behaviour, which are [9]:

(a) Failures related to the controller (physical controllers)

(b) Unsafe Control input

(c) Inadequate control algorithm

(d) Inadequate Process model

Unsafe control actions could occur in situations wherein inadequate information or feedback is provided to controllers. Feedback comes from controlled processes via sensors and other components. Incomplete or missing feedback could lead to an incorrect control output being executed by the controller, which inadvertently leads to a hazard. Thus, it is vital to identify scenarios concerning the following [9]:

(a) Feedback or information is not received.

(b) Inadequate feedback is received.

2. **Why would control actions be improperly executed or not executed, leading to hazards?**

   As mentioned earlier in Section 2.1.3, hazards are caused by unsafe control actions. Furthermore, hazards can also be caused by control actions which have not been executed in the manner they are required to be. Thus, scenarios must be created considering factors affecting the control path as well as the controlled process. The control path is responsible for transferring control actions from the controller to the controlled process. The defined control path could include a simple actuator or may involve a series of actuators, thus there could be a simple or complex path. Irrespective of the type of control path used, issues in the control path leading to improper control action execution must be identified. Scenarios related to control path may include:

   (a) **Control action is not executed.**

   (b) **Control action is improperly executed.**

   A control action may be properly executed by the controller, however it may not be transferred or applied correctly to the controlled process. The control action issued by one controller could be overridden by another controller. Scenarios related to problems faced in the controlled process are:

   (a) **Control action is not executed.**
   The control action is received by the controlled process, but the process does not respond to the input.

   (b) **Control action improperly executed.**
   There could be two possibilities in this case:

      i. The control action is received by the controlled process but the controlled process responds incorrectly.

      ii. The controlled process does not receive the control action but responds as if a control action has been received by it

Therefore, by following the above mentioned process, loss scenarios are obtained for a given system. The identification of loss scenarios concludes the steps undertaken in STPA.

**Categorisation of loss scenarios**

The process of listing out loss scenarios results in the estimation of a large pool of cases. However, each case in the pool may not have the same level of importance and thus can be considered as a lower priority. Thus, the loss scenarios must be ranked based on their severity and probability of exposure [19]. The severity represents an estimate of the potential harm in a particular driving situation, while the probability of exposure is determined by the possibility of occurrence of the

situation [20]. The levels of severity are set in a range between 0 and 3, where 0 is the least severe and 3 is the most severe. A similar approach has been considered for the probability of exposure, where 0 is the least probable and 3 is the highly probable case. The product of the two sets of values, as illustrated in Equation 2.1, are used for estimating the priority of any given scenario [21]. Scenarios, which have a product of severity and exposure equal to 9, are considered as high priority scenarios.

$$Priority\ (p)\ =\ Probability\ of\ exposure(E)\ *\ Severity(S) \tag{2.1}$$

## 2.2 Selection of autonomous driving simulator

A highly reliable autonomous driving vehicle requires testing of autonomous characteristics in every possible scenario [22]. The design, implementation, and testing of vehicles in a wide range of use cases, in realistic traffic and weather conditions are not only costly but also time-consuming [22]. Replicating a given worst-case environment or condition to test an autonomous driving vehicle is a challenging task. A suitable solution for autonomous driving software testing is a virtual platform in the form of an autonomous driving simulator.

Autonomous driving simulators provide an environment within which different functions of the AD system can be deployed and tested [22]. Software deployment of such systems can be performed on such platforms. An autonomous driving simulator is used for testing an autonomous vehicle within a defined virtual environment [23]. Key performance indicators are used to compare the performance of the vehicle in different scenarios. To select an appropriate simulator, a criterion for evaluation has been drawn, which are as follows:

1. **License Mode:** The simulator will have a license that is either open-source or commercially available. If the license is commercially available, the license must be available within the list of software provided by the university.

2. **Operating system:** The simulator can be executed on either Linux or Windows OS. This criterion is informative and will not be used in the decision-making process.

3. **Customization of Simulation environment/ World:** Making modifications to the existing simulation environment or world is a hard requirement. These modifications would aid in simulating varying scenarios. This includes controlling the weather, traffic, and pedestrians. It also comprises the choices of vehicle assets that could be used in simulations. The following features of the simulation environment to be investigated are:

   (a) Importing new worlds
   (b) Varying traffic and weather conditions
   (c) Control over pedestrians
   (d) Multiple vehicle choices

4. **Setup and Execution time for simulation:** Setup time refers to the time taken for setting up the simulation. This includes initiating the build process for the simulator and inputting the parameters for simulations. Execution times refer to the time taken by the simulator for running a simulation given a set of parameters.

5. **Customization of control strategies:** The simulation environment must provide scope for adjusting control algorithms utilized behind the functioning of autonomous driving systems. Different types of control strategies must be employed to demonstrate the impact of maps on varying strategies.

6. **Production of videos:** The simulator must serve as a visual aid to further the understanding of the objective of the simulation. By producing videos of good quality, we can demonstrate the results of the analysis visually.

Simulators have been reviewed by TomTom based on a set of hard and soft requirements [24]. The simulator would be used for testing the self-developed stack present in its self-driving unit. Hard requirements are mainly associated with providing simulation data, ROS interfacing, performance on currently utilized hardware, the possibility of executing regression tests, and lastly physical model of the vehicle and the world. Soft requirements are related to the quality of vehicle, world, and weather and daytime simulation, usability, collision detection, and avoidance, and built-in ROS-interfaces. The price of the license, if commercially available, is also a factor that has been considered. Since the requirements coincide with those which are needed for the given application, the results of this review can be utilized.

Simulators that have been reviewed by TomTom are CARLA [25], Gazebo [26], Airsim [27], DeepDrive [28], Udacity Self-Driving car simulator [29], GTA V [30], AutonoVi [31], CarSim [32], IPG Carmaker [33], V-Rep, aiSim [34], VTD [35], PreScan [36], Webots [37] and LG Simulator [38]. Majority of the mentioned simulators are commercially available whose licenses aren't owned either by TomTom or TU/e, thus are out of consideration. Simulators that have been shortlisted by TomTom are as follows:

1. CARLA
2. Webots
3. LGVSL

To finalize an autonomous driving simulator for this project, an evaluation of the shortlisted options must be conducted. Table 2.1 describes the comparison conducted on the simulators available based on the criterion mentioned above.

Table 2.1: Comparison of autonomous driving simulators.

| SR.No | Criterion | CARLA | TomTom's Software | LG Simulator | Webots |
|---|---|---|---|---|---|
| 1. | License mode | Open Source | Owned by TomTom | - Open Source<br>- Limited documentation | - Open Source<br>- Limited documentation |
| 2. | Operating system | Windows and Linux | Linux | Windows and Linux | Windows (Partially) and Linux |
| 3. | Customization of Simulation Environment/World | | | | |
| 3(a). | Importing new worlds | Using OpenDrive standards, worlds can be made in RoadRunner | Directly imported through AutoStream plugin | Formats supported are: Lanelet2, Apollo 5.0 HD, and OpenDRIVE Map | OpenStreet map editor can be used for editing worlds. |
| 3(b). | Control over traffic and weather conditions | Yes | Traffic can be updated, weather remains the same | Yes | Yes |
| 3(c). | Control over Pedestrian/ obstacles | Yes | Yes | Yes | No pedestrians are present |
| 3(d). | Choices of Vehicle | 31 vehicles available | Single vehicle (Toyota Prius) | 5 vehicles | 3 types of vehicles based on propulsion (Combustion, Electric, and Hybrid) |
| 4 | Setup and Execution times for simulation | Low setup and execution times. | Linux OS and 8 core processor are minimum requirements. | Low setup and execution times. | Low setup and execution times since system requirements are relaxed in comparison to other simulators. |
| 5 | Customization of control strategies | They can be programmed using C++, python API. | They can be programmed with ROS. | They can be programmed using C++, python API. | They can be programmed using C, C++, Python, MATLAB, or ROS. |
| 6 | Production of videos | Unreal engine produces high-quality visuals. | Good quality videos can be produced using the given API. | Unity engine produces high-quality visuals. | Good quality videos can be produced using the given API |

From Table 2.1, we can confirm the selection of CARLA. CARLA will be used for conducting autonomous driving simulations. An autonomous vehicle will be controlled in the CARLA in a defined environment and tests will be conducted on the same.

## 2.3 Introduction to CARLA

CARLA is an open-source autonomous driving simulator created to support development, training, and validation of autonomous driving systems [25]. Furthermore, CARLA provides open-source digital assets such as vehicles, buildings, and layouts of worlds which can be used freely by the developers. The simulation environment supports flexible specification of sensor suites, environmental conditions, full control over all static and dynamic actors, and maps generation[25].

CARLA is built on Unreal Engine to run the simulation environment. It uses the OpenDrive standard (1.4) to define roads and urban settings. The simulator consists of a scalable client-server architecture [39]. The server is responsible for tasks conducted within the simulation environment such as sensor rendering, computation of physics, and updates on the world-state. The client side consists of multiple client modules which are responsible for controlling the logic of actors on the scene and setting the world conditions. The client-server communication is achieved through the CARLA API (Python or C++) [40]. The defined structure is illustrated in Figure 2.4.

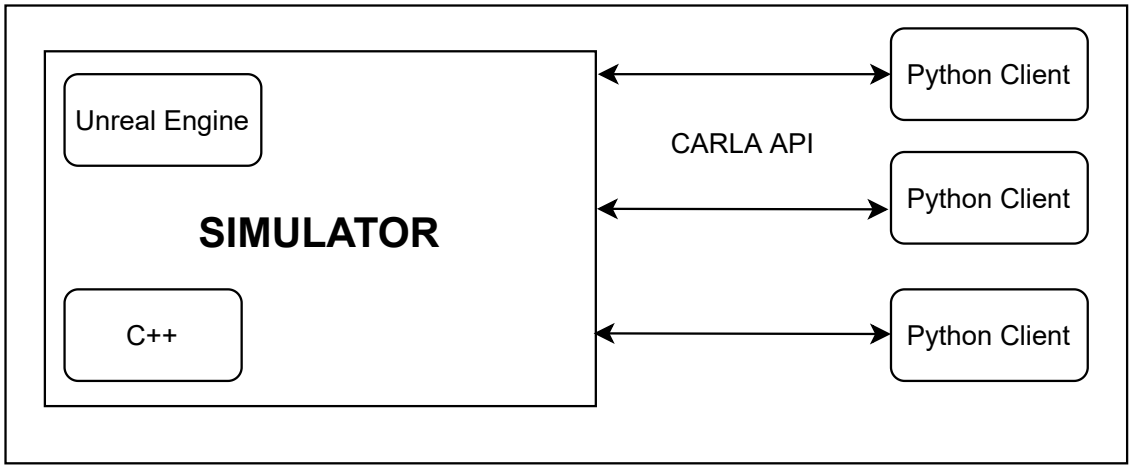


Figure 2.4: Architecture of CARLA.

In the following sub-sections, the various elements of the autonomous driving simulator are presented. The process of connecting a world, followed by the spawning of actors such as sensors and vehicles, has been described.

### 2.3.1 Initialization of world in CARLA

As discussed in the previous section, CARLA is built on an Unreal Engine foundation [25]. This functions as the server in the defined architecture. To connect to the server side of the simulator, a "client" object must be created. This is done by providing an IP address and the port number of a running instance to the CARLA API. Following the creation of the object, a timeout must be set to limit networking operations and to ensure the client is not blocked forever. A world object is created using the defined client object to retrieve the existing world.

The world in CARLA represents the loaded map and contains the required functions needed for converting a blueprint into a living actor. Access to the road map and functions for changing the weather conditions have also been provided using the CARLA API. The world is modelled using Unreal Engine. The loaded map includes both a 3D model of a town and its road definitions. The map is made from an OpenDrive file which describes the road layout [41]. The CARLA API enables high-level querying for navigating through the roads defined in the map.

Currently, CARLA provides 8 different maps, which corresponds to 8 different worlds [42]. The blueprint for each town is available in CARLA's blueprint library. Different maps can be loaded in the server by changing the input to the name of the town needed using the CARLA API [40]. However, if a different map is loaded, the server must be rebooted and created from scratch.

The map defined in the blueprint contains features such as landmarks, junctions, lanes, and waypoints. Landmarks refer to the traffic signs and traffic lights which have been defined in the OpenDrive file [42]. The orientation, type and location can be queried using the Landmark object. The properties of lanes such as the type, color, lane width, and the thickness of the marking can be accessed using the LaneMarking object. A waypoint in CARLA refers to a 3D-directed point in the world. They contain some information regarding the lane containing that point such as the left and right lane markings. Using these features, a vehicle is navigated through a defined route in a world in CARLA.

### 2.3.2 Selection and control of vehicles

Following the creation of the world, actors must be spawned in the world to perform different functions [40]. There are multiple kinds of actors that can be spawned in the world such as sensors, spectators, vehicles and walkers. A vehicle is a special type of actor in CARLA. This actor is used in simulating the physics of a wheeled vehicle. In CARLA, there are 2-wheeled and 4-wheeled vehicles. 2-Wheeled vehicles consist of bikes and motorcycles models such as Yamaha and Harley Davidson [43]. 4-Wheeled vehicles consist of cars manufactured by OEMs such as Audi, Chevrolet, Dodge, Lincoln, Tesla, and Mercedes [43].

The process for spawning a vehicle in a map is similar to that used for an actor in CARLA. The model for the vehicle must be filtered out from the blueprint library [40]. Once the model has been selected, a given spawning location and vehicle orientation must be provided. The vehicle is spawned using the selected model and its' spawning location. The process used for spawning a vehicle has been highlighted in Figure 2.5.

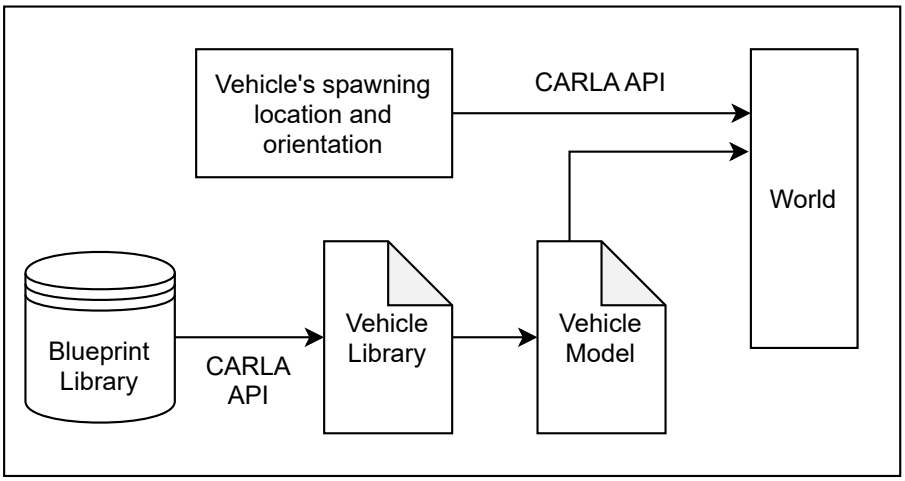


Figure 2.5: Process for vehicle spawning in CARLA.

A vehicle is controlled in CARLA by providing throttle, brake, and steer values. The brake, throttle and steer values are float by type. The handbrake and reverse inputs are boolean by type. The vehicle is set to reverse by changing the state of the boolean variable. Furthermore, CARLA provides a hard-coded autopilot mode, which is a of boolean type [40]. By enabling autopilot mode, the vehicle is controlled by the defined traffic manager. These values must be taken into consideration during the design of a controller for lateral and longitudinal movement. The properties utilised for controlling a vehicle and their respective ranges are presented in Table 2.2.

Table 2.2: List of vehicle properties in CARLA.

| Vehicle property | Type | Range |
|---|---|---|
| Throttle | Float | [-1,1] |
| Brake | Float | [0,1] |
| Steer | Float | [-1,1] |
| HandBrake | Bool | True/False |
| Reverse | Bool | True/False |
| Autopilot | Bool | True/False |

### 2.3.3 Route planning in CARLA

To drive a vehicle on a defined route, a set of waypoints must be used. A waypoint in CARLA refers to a 3D directed point. The waypoints are listed using the global coordinate system of the map. The global coordinate system for the map is presented in Figure A.2. They can be retrieved from the OpenDrive file which is used for the creation of the simulation environment. Waypoints have a set of methods to connect with other waypoints and form a road flow [40]. These points are created in accordance with the road rules, thereby ensuring the vehicle can track a legal path.

Before a route can be generated, an instance of the map must be created. The map is generated in an XODR format, which is then parsed to the created map object. The user must specify the initial location and final destination [42]. The two sets of coordinates received will then be used in the route generation process.

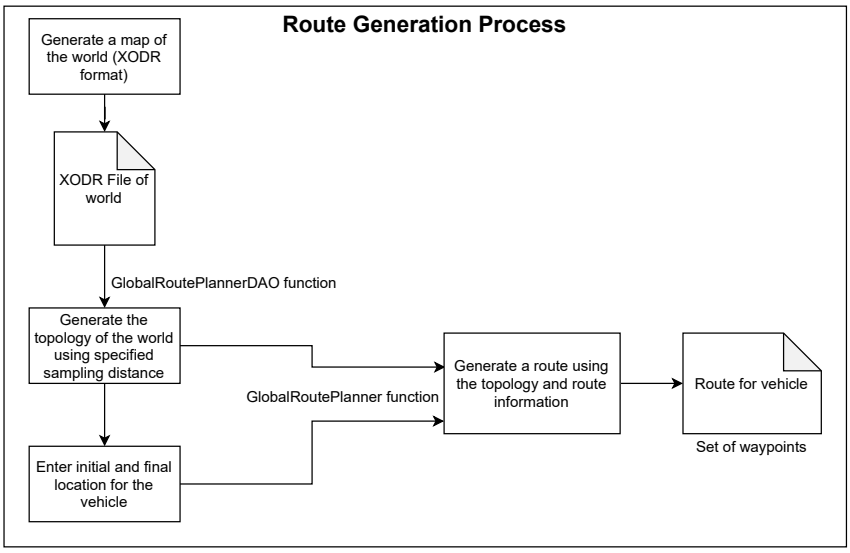


Figure 2.6: Process of route generation used in CARLA.

The route generation process has been carried out using the GlobalRoutePlanner provided in the CARLA API [40]. Before using the above defined function, data must be fetched from the server. This is achieved by executing the GlobalRoutePlannerDAO function [40]. This function retrieves topology from the server in the form of a list of road segment as pairs of waypoint objects. The sampling distance, the distance between two defined waypoints, must be specified as an input to this function. Depending on the level of granularity required, the user can request a data set with small step size (order of centimeters) or a larger step size (order of meters). The data gathered from this step is then used for tracing a route from the initial to final location. The function yields a set of waypoints which can be used for navigating the vehicle. The process is illustrated in Figure 2.6.

### 2.3.4 Selection of path following controller

This section presents the selection of the controller used for driving the vehicle autonomously within the defined world in CARLA. The controller calculates the error between the current path and the trajectory to be followed to produce a control action. The control action consists of a normalised steering angle which is given as an input to drive the vehicle. The control models which are used extensively for the lateral control of an autonomous vehicle are as follows:

1. **Pure pursuit controller:** Pure Pursuit method is based on the geometric model of the vehicle. The pure pursuit approach is a method of geometrically determining the curvature that will drive the vehicle to a chosen path point, termed the goal point. This goal point is a point on the path that is one look-ahead distance from the current vehicle position [44].

2. **Stanley controller:** This method comes from the robot system Stanley that won the DARPA Grand Challenge [45]. This method defines the steering control law as a non-linear function of the cross-track error (taken from front axle here), that is, the distance between the front axle of the vehicle and the nearest point on the path.

3. **Proportional-Integral-Derivative Controller (PID)**: It is one of the most popular control loop feedback mechanism for implementing cruise control and lateral control of a vehicle. The process variable considered is steering angle and the manipulated variable as steering input (for both simulated vehicle and realistic vehicle). Thus, according to PID control theory, the error term, that is a difference between the set point and process variable, controls the steering input of the vehicle proportionally.

The simulation will be conducted at low speeds at which the Pure Pursuit and Stanley controllers would provide optimal results [46]. A PID steering controller is not recommended at low speeds due to high sensitivity to gains [46]. It is difficult to tune the look-ahead distance for PID controller [47]. Furthermore, the level of performance of a PID controller is dependent on the tuning of the parameters, which could serve as a time-consuming process [48]. For a pure pursuit controller, the starting point, whether it starts on the path or off it, plays an instrumental role [47]. The tuning of a Pure Pursuit controller can be accomplished by varying a single parameter, look-ahead distance [46]. After careful consideration based on given time constraints and the discussion above, we can conclude on the selection of a Pure Pursuit controller for the lateral control of the vehicle.

The implementation of the pure pursuit algorithm is a simple task. Flowchart indicating the algorithm is presented in Figure A.3, which has been attached to Appendix in Section A.3. The algorithm has the following steps [44]:

1. ***Determine the current location of the vehicle:*** The position of the vehicle is reported using a Global Positioning sensor (GPS) mounted on the vehicle spawned in CARLA. An Inertial Measurement unit (IMU) has also been attached to provide information regarding the heading of the vehicle

2. ***Find a path closest to the vehicle:*** In a geometrical sense, the point must be located within one instance of the defined look-ahead distance. If there are multiple coordinates within the defined interval, the closest point to the vehicle must be selected. The next point must be selected in the second instance of the look-ahead distance.

3. ***Find the goal point:*** The goal point is estimated while moving on the defined path and by calculating the distance between the vehicle' current location and the next point estimated in the previous step.

4. ***Transform the goal point to vehicle coordinates:*** On finding the goal point, the coordinate must be converted to the vehicle's local coordinate system. The location of the path point are available in global coordinates, thus must be converted. The vehicle's coordinate system is illustrated in Figure A.1, which is placed in Section A.1 of the Appendix.

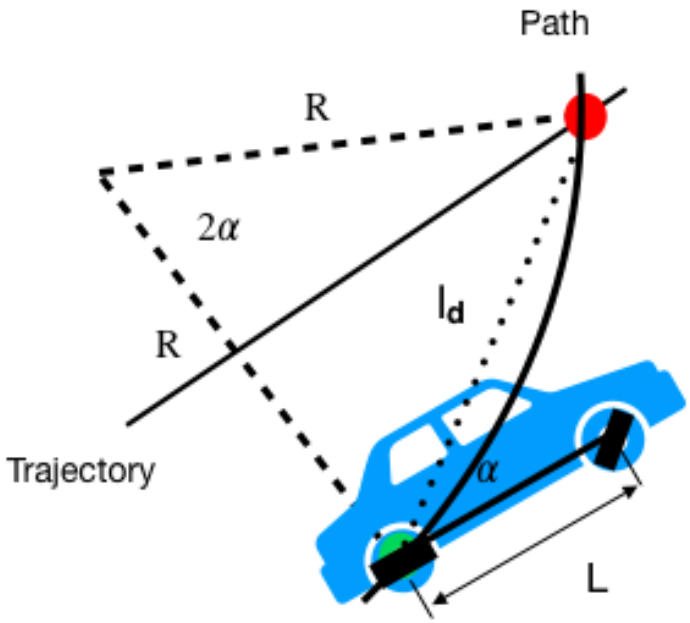


Figure 2.7: Diagram indicating calculation of steering angles for a vehicle.

5. *Calculate the curvature needed to meet the goal point and estimate the steering angle needed:* The desire vehicle curvature is estimated using the equation below. Figure 2.7 represents the calculation of the steering angle for the vehicle. The curvature obtained is translated to a steering angle for the front wheels [49].

$$\delta = arctan(\frac{2Lsin\alpha}{l_d}) \tag{2.2}$$

where:
$\delta$ = Steering angle of the vehicle
$L_d$ = Wheelbase of the vehicle
$\alpha$ = Difference between vehicle's body heading and look-ahead line

6. ***Normalise the steering angle based on the required input to be given to the vehicle model:*** As explained in Section 2.3.2, the steering input provided to the vehicle model must be within -1 and 1. Thus, the steering output from the controller must be normalised using the maximum steering angle available for each vehicle model. Cases could be encountered wherein the steering angle generated is higher than the maximum steering angle of the vehicle. In these cases, the steering angle must be clipped to the maximum available steering angle. Once the angle has been clipped and then normalised, it can be passed as an input to the algorithm, which then generates a steering input in the required intervals.

7. ***Update the vehicle's position:*** The steering input is provided to the vehicle, and it makes the required manoeuvre using the new inputs. This results in a change of vehicle position which is then fed back to the system and the algorithm repeats itself.

The implementation of the above-mentioned algorithm has been split into two different functions, with one function tailored for the estimation of the goal point and the second one for control output generation. The functions have been attached to the Appendix in Section C.1.

# Chapter 3

# Methodology

In this chapter, the methodology which has been used in the thesis is presented. The previous chapter described how can STPA be applied to a given system. The application of STPA to the defined problem statement is described. This is followed by the description of the tasks to be performed in the CARLA, which includes the processes involved in conducting tests and gathering information for post-processing.

## 3.1 System Theoretic Process Analysis of automated driving

The processes involved in performing System Theoretic Process Analysis (STPA) on a given system has been documented in Section 2.1. STPA was performed on a level 2 or 3 autonomous vehicle [17]. STPA was conducted on the vehicle from the point of view of the map manufacturer, TomTom [1]. The safety analysis starts by defining the system, followed by its boundaries. System definition includes the formulation of the high-level system architecture which was used to depict the communication with external systems. This was followed by the identification of stakeholders and their respective values, which was inverted to create a list of losses. The analysis was aimed towards eliminating the losses for the defined stakeholders.

After identifying the list of losses, hazards and the system constraints, the system was modelled in the form of control structures. The high-level abstraction of the system made in the previous step was broken down further to form multiple low-level control structures. Functions and feedback between sub-systems and systems was also modelled in this step. Control actions performed by each subsystem and their respective issuer was identified in each control structure. The list of control actions were used for identifying unsafe control actions based on the criterion mentioned in Section 2.1.3. Unsafe control actions were inverted to formulate controller constraints to prevent the occurrence of unsafe control actions in a given subsystem.

STPA was aimed at preventing or minimising the losses from occurring in the system, therefore the context behind the occurrence of an unsafe control action must be identified [9]. This was done by identifying loss scenarios for the given pool of unsafe control actions. Since the analysis was performed from the point of view of the map manufacturer, TomTom, control actions related to the map features were assigned a higher priority. Unsafe control actions identified from the categorised control actions were used for the scenario identification process. Scenarios are listed using the defined categories mentioned in Section 2.1.4. The process flow of the results obtained is illustrated in Figure 3.1.

Lastly, the list of loss scenarios was categorised based on the respective severity and probability of exposure [20]. The product of the severity and probability of exposure was used to ascertain the priority of each scenario. A root cause analysis of the list of high priority scenarios was performed to understand the context, entities, and the different triggers present in the scenario

[50]. Furthermore, the map feature and its respective key performance indicators which would be required by the vehicle in each scenario, to prevent the occurrence of a hazard, were identified.

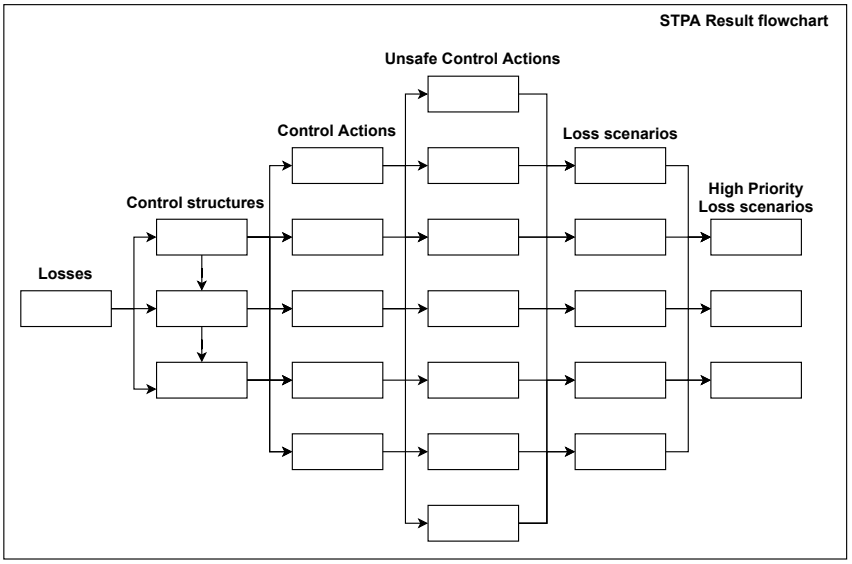


Figure 3.1: Flow of results of STPA.

## 3.2 Validation of loss scenarios

Loss scenarios were identified for the autonomous driving vehicle to describe the causal factors that can lead to the unsafe control actions and to hazards [9]. Loss scenarios were categorised based on their respective severity and probability of exposures. The scenarios with the highest levels of severity and probability of exposure were labelled as high priority. However, the priorities calculated for each scenario in the list of scenarios must be verified. By having a validated list of scenarios, the priority placed on a given scenario in the list can be justified. The verification was conducted using customer feedback, i.e, clients who purchase TomTom's products. Each client has their own set of requirements which are met by TomTom. Each client may not necessarily use the product in the same manner, thus resulting in a diverse list of use cases.

The validation process was marked by pooling together the use cases provided by each customer.This has been illustrated in Figure 3.2. This resulted in the formation of two sets of scenarios. Scenarios are validated by either finding a match between the two lists or by finding commonality in the description of the scenarios. Commonality in scenarios refers to a common map feature such as lane markings between scenarios, which has been highlighted in both lists of scenarios. Two scenarios concerning incorrect traffic sign positioning were considered to be a match since they had commonality with respect to the map feature, traffic sign. The definition of the scenarios may be different. The lists were placed alongside each other and matches with respect to map features and scenario description were found.

In addition to the customer feedback, TomTom's measurement data, which is used for producing maps, was also used in the validation of loss scenarios. Sensor data from the Mobile Mapping (MoMa) vehicle was coupled with the High Definition (HD) map data on TomTom's proprietary software. This software was used for identifying scenarios resulting from the irregularities in the streamed map features or the sensor data gathered from the data. Since the MoMa vehicle has covered a large proportion of the available roads, large number of use cases on different types of roads were covered. The process of validation is illustrated in Figure B.6.

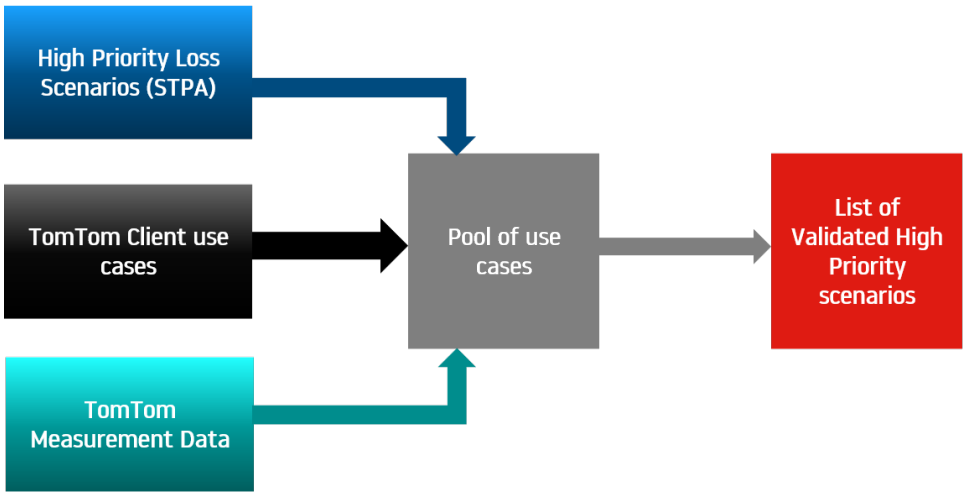


Figure 3.2: Sources of data used for scenario validation.

## 3.3 Simulation of uncertainty in map in CARLA

The results of the safety analysis yielded a list of scenarios in which the vehicle performs an unsafe control action, which subsequently leads to a hazard and a loss. The map feature involved in each high priority scenario was determined. Furthermore, a study on the importance of the map in a worst-case environment was conducted by performing tests. The impact of noise in a selected map feature on the positional accuracy of the vehicle in a defined world was estimated. This study was performed in a simulation environment, CARLA.

A high definition (HD) map in CARLA contains both a 3D model of a town and its road definition [40]. In a simulation environment, a map is an identical replica of the physical world. However, when a map is built using ground truth data, errors are generated whilst replicating those features in a model. The vehicle uses the fusion of map data and sensor data to make decisions whilst driving in autonomous mode. Thus, errors in the map could play a fundamental role in the decision-making process for the vehicle. The impact of these errors on the vehicle's lateral control must be ascertained. The quality of the map must be ensured by setting a defined accuracy during the production process. The accuracy of the map is classified in two categories, which are as follows:

1. **Absolute accuracy:** Closeness of reported position values to the values accepted as or being true [51].

2. **Relative accuracy:** Closeness of relative positions of features in a data set to their respective relative positions accepted as or being true [51].

The requirements set on the accuracy of the map can be analysed by studying the impact of varying levels of accuracies on the vehicle's lateral performance. The feature of the high definition (HD) map in CARLA taken into consideration are waypoints [40]. The set of waypoints generated from a high definition (HD) map in CARLA can be considered to be equivalent to the lane geometry feature of TomTom's map. Since waypoints in CARLA are generated in the center of the lane, lines created using these points can be compared to the lane centerline. The route generated in CARLA yields a list of waypoints. The identified waypoints can be printed in the world using the CARLA API [40].

The set of waypoints generated in CARLA are present at the center of the lane, which would represent the ideal case. To depict a case of a map made from ground truth data, noise was injected in the waypoints. The noise injection process is highlighted in Figure 3.5.

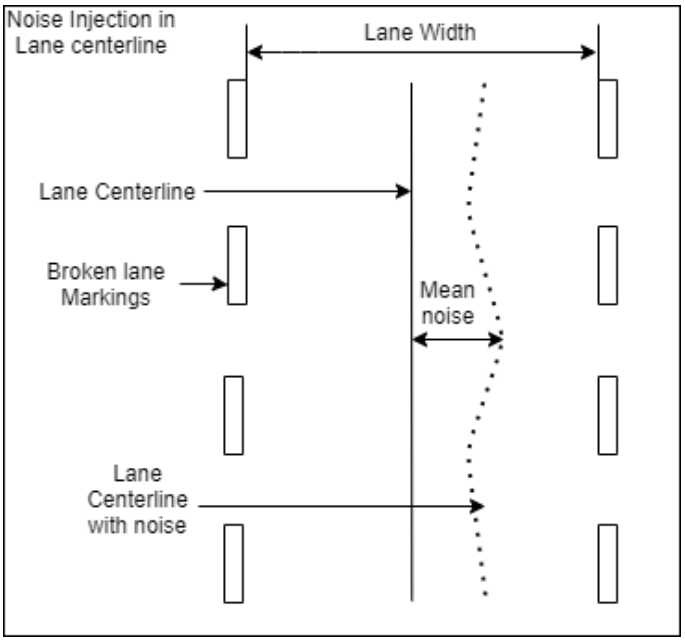


Figure 3.3: Top view of lane centerlines with Gaussian Noise.

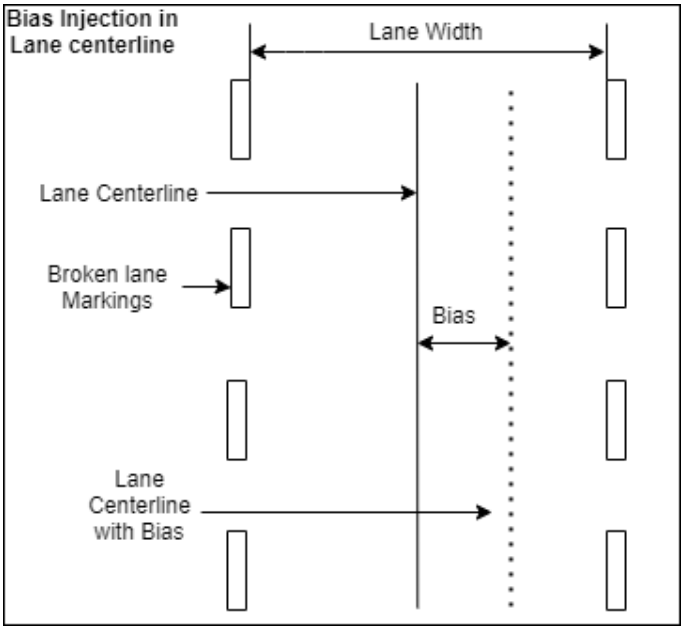


Figure 3.4: Top view of lane centerlines with a bias only.

An OpenDrive file is used for creating the world in CARLA [42] [41]. For the given application, pre-defined OpenDrive files provided in the build of CARLA were utilised. This OpenDrive file was used for the generation of waypoints, which was utilised in route generation. A vehicle model was selected from the Blueprint library in a parallel manner, which was prepared for simulation. The generated noise signal was injected in the route used for path tracking by the vehicle. The noise signal has a mean noise and jitter component that was varied as per the parameters set for the given simulation.

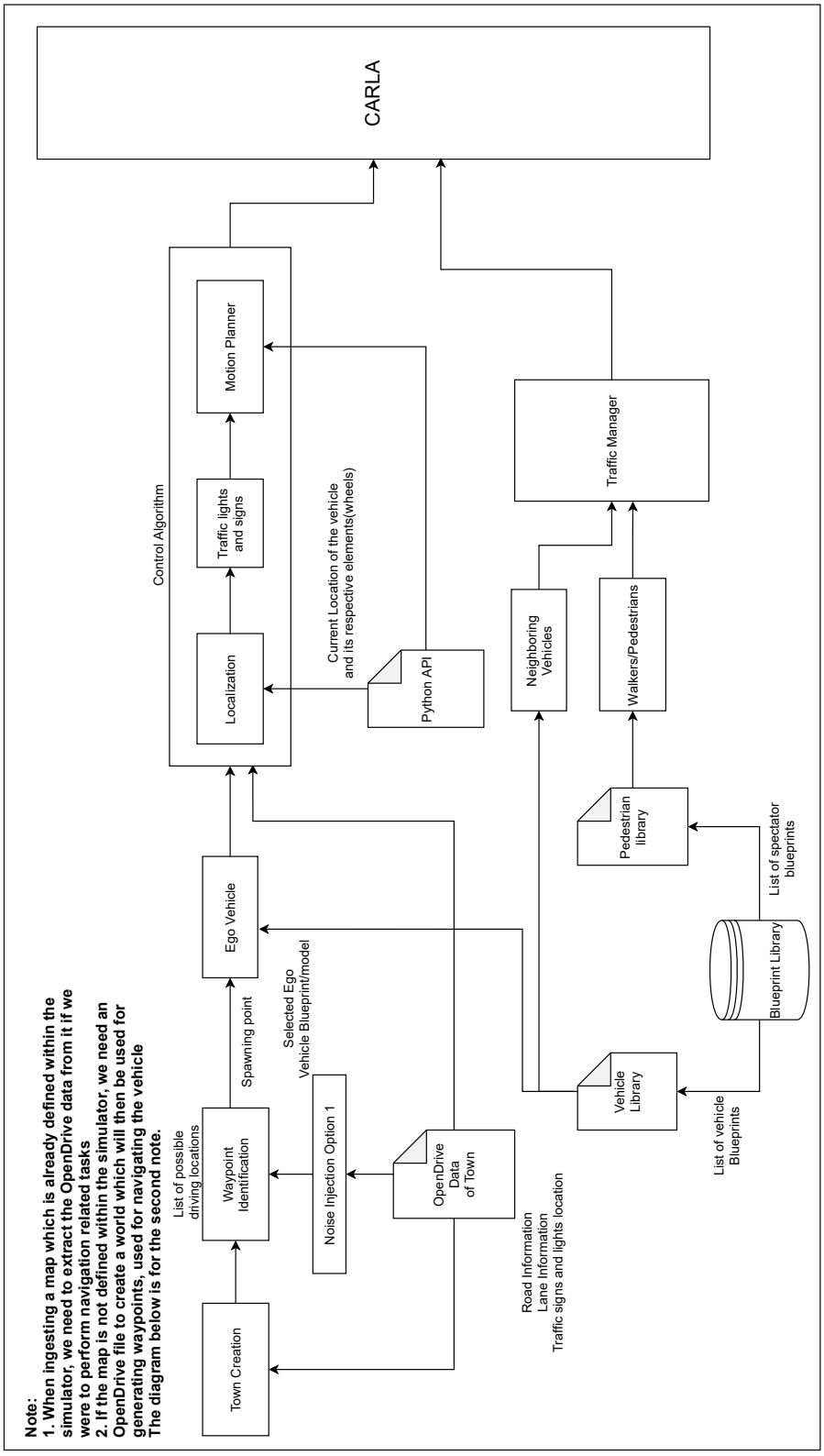


Figure 3.5: Block diagram representing simulation in CARLA.

By varying the magnitude of noise injected in the waypoints, maps of different quality levels were generated. The injected noise signal comprised a bias and jitter component. The bias refers to the mean of the distribution of noise. The signal will be generated using a normal distribution. Figures 3.3 and 3.4 represent the top view of lane centerlines in which a gaussian noise distribution with and without a standard deviation have been injected respectively. The function created for injecting noise into the list of waypoints has been attached to the appendix in Section C.2.

CARLA can be used for performing multiple types of autonomous driving simulations [25]. Since the focus of the project was solely on the impact of the map on the functional safety of an automated driving vehicle, certain assumptions regarding the simulation environment were made to limit the scope. The assumptions were decided based on the list of scenarios obtained from the STPA analysis conducted. Furthermore, the assumptions should aid in the depiction of a worst-case condition for the autonomous vehicle in a defined environment. The set of assumptions defined for the simulation are as follows:

1. Vehicle is operating/driving in automated driving mode.
2. Vehicle is reliant only on GPS, IMU, and map data for manoeuvring a given route.
3. The vehicle has perfect localization, minimal errors in the positional estimate made by the GPS and IMU.
4. Camera and LiDAR have been disabled to portray a worst-case environment for the vehicle's automated driving (AD) system.

Before conducting simulations in CARLA, a criteria for evaluating each case simulated must be defined. The criteria was defined in the form of key performance indicators (KPIs). They were used to estimate and evaluate the impact of uncertainty on the safety of the vehicle in different scenarios. In a given worst-case situation, the vehicle can be considered to be in a safe state if it not only follows the system-generated path with minimal errors, but also does not exit the currently occupied lane. Using this information, the defined KPIs are presented below. The key performance indicators are presented in Figure 3.6. The key performance indicators (KPI's) are as follows:

1. **Mean absolute error (MAE):**

$$MAE = Mean\ (Vehicle's\ location\ -\ Coordinates\ of\ the\ route\ w/o\ bias) \tag{3.1}$$

   The error generated in tracking a path, which has an element of bias introduced in it. The error is estimated by taking the average of the difference of the vehicle's location with respect to the original route (bias=0). Mean absolute error is presented graphically in Figure 3.6.

2. **Sensitivity:**

$$Sensitivity = \frac{\Delta MAE}{\Delta Bias} \tag{3.2}$$

   Sensitivity refers to the change of mean absolute error with respect to the change in mean noise injected in the map feature. 1 unit of sensitivity is equal to the ratio of 1 unit of change in mean absolute error to a change in a single unit of mean noise. This was used to understand the effectiveness of the noise injected on the mean absolute error for a given case.

3. **Lane invasions:**
   It refers to the number of occasions the vehicle cuts a lane. The function prints the number of times the vehicle cuts a given lane and the type of lane it cuts into. This parameter was used to compare performance of the vehicle in difference cases of bias. Furthermore, the positions at which lanes are cut by the vehicle are also observed.

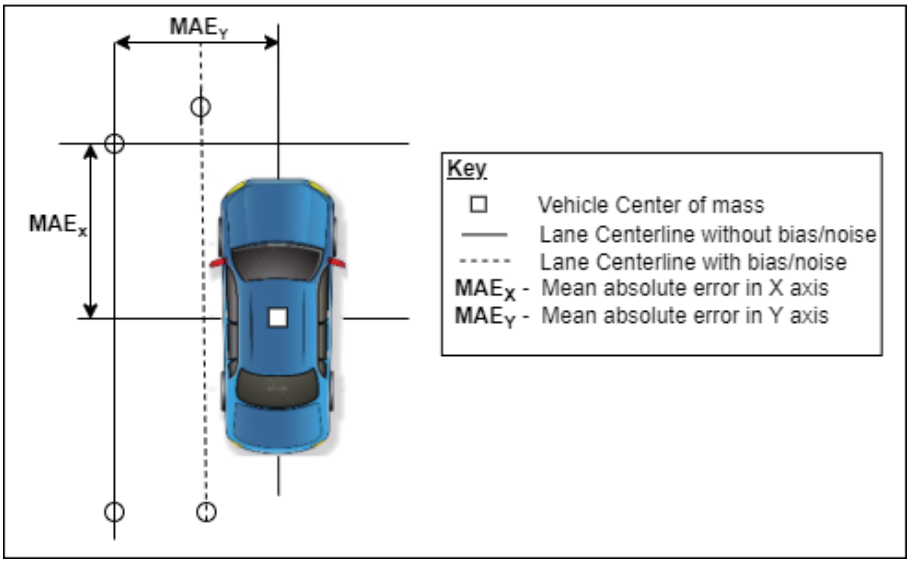


Figure 3.6: Key Performance Indicators (KPI) for simulation in CARLA.

Simulations were conducted using different sets of parameters and data concerning the above-mentioned KPI's was gathered. Following the completion of the tests, the data gathered was analysed, and a conclusion was drawn regarding the behaviour of the KPI's with respect to the parameters. Thus, the above-stated methodology was applied to the selected SAE Level 2/3 automated driving vehicle.

# Chapter 4

# Vehicle-level Safety analysis

In this chapter, the results of the STPA applied to an automated driving vehicle using maps is discussed. The results is presented from the point of view of the map provider and focus is placed upon the usage of maps.

## 4.1 Introduction to selected automated driving system

The vehicle under consideration is a Level 2/3 as per SAE standards of automation [17]. The vehicle has a human machine interface (HMI) which is used by the driver for triggering different modes of operation. The emergency brakes can also be activated using the HMI. The driver must start the vehicle and drive it manually. The vehicle's software system begins to run when the vehicle is in motion. This ensures obstacles can be detected on the fly, thus enabling the engagement of the collision avoidance system when required. The actuators for automated driving will be activated once the mode of operation has been set to automated by the driver. The driver can switch the state of the vehicle only after a set time period (mode transition) has lapsed. In the event of malfunctioning subsystems, take-over requests (TORs) will be issued to the driver, which he/she must accept. The following sensors are mounted on the vehicle:

1. Inertial measurement unit (IMU) - POS LV 125
2. Global navigation satellite system (GNSS) - u-blox GNSS
3. Camera - Sekonix SF3325 and SF3324
4. LiDAR - Ibeo Lux 4L and Velodyne VLP 16
5. Radar - Smart Micro UMRR-8F

## 4.2 Vehicle-Level losses and initial hazards

STPA begins by identifying the stakeholders present in the system. This is followed by listing out the goals of each stakeholder, which were then inverted to formulate losses. The stakeholders present in the system are as follows:

1. Driver and Passengers (users) of the Autonomous Vehicle (AV)
2. Automated driving Vehicle Manufacturers (OEMs)
3. Map Providers (TomTom)
4. Government and other legislating bodies (Road Safety organisations)

Using the list of stakeholders, losses were ascertained. The goal of STPA was to mitigate the losses formulated. The losses are listed in Table 4.1.

Table 4.1: List of losses identified from Step I of STPA.

| **Loss ID** | **Loss definition** |
|---|---|
| L1 | Loss of Life or injury |
| L2 | Loss of or Damage caused to road infrastructure |
| L3 | Loss of vehicle type approval/certification |
| L4 | Loss of longitudinal control of vehicle |
| L5 | Loss of lateral control of vehicle |
| L6 | Loss in completeness of maps |
| L7 | Loss in positional accuracy of maps |
| L8 | Loss in thematic accuracy of maps |
| L9 | Loss in logical consistency of maps |
| L10 | Losses in timeliness in supply of maps |
| L11 | Lack of feedback to map providers |

The identified losses were used for defining hazards. In this analysis, the system consists of the automated driving vehicle. The vehicle utilizes an automated driving system and maps to automate the driving process. The scope of this project was limited to these elements. Human operators of the system refer to the driver and passengers in the vehicle. Furthermore, the OEMs responsible for manufacturing the vehicle and the map providers were also considered. Before establishing the system-level hazards, system-level states had to be set. The states of the automated driving system are as follows:

1. The vehicle can either be in motion or at rest (parked).
2. The vehicle in motion can be in three possible modes of operation: Automated, manual or transitional.

Using these states, system-level hazards were identified. Each hazard has been further broken down into sub-hazards which could be caused by the system. The identification of each loss caused by the hazard has been mentioned after defining the hazard. This has been done to ensure traceability of losses with respect to the hazards. The hazards have been listed in Table 4.2. In the table below, AV stands for automated driving vehicle.

Table 4.2: List of hazards identified in Step I of STPA.

| **Hazard ID** | **Hazard Definition** |
|---|---|
| H1 | The AV violates spacing requirements on the road. [L 1,2,4] |
| H1.1 | Excessive acceleration is done by the AV whilst navigating through traffic. |
| H1.2 | Insufficient deceleration is done when approaching another car in traffic. |
| H2 | The vehicle engages autonomous mode in restricted areas of the map. [L 1,2,3,4,5,6] |
| H3 | The AV follows the wrong trajectory. [L 1,2,5,7,8,10,11] |
| H3.1 | The AV follows the defined path based on map data. |
| H3.2 | The AV follows a path based on sensor observations. |
| H4 | The vehicle cannot establish a working data connection. [L 10,11] |
| H5 | The AV initiates a wrong take over request (TOR). [L 1,2,3,4,5,6] |
| H5.1 | The vehicle in autonomous mode issues a wrong TOR. |
| H5.2 | The driver unexpectedly assumes control of the car. |
| H6 | The AV cannot visualize the surrounding environment. [L 1,2,3,4,5,11] |
| H6.1 | The AV is unable to make any observations. |

Table 4.2: List of hazards identified in Step I of STPA.

| Hazard ID | Hazard Definition |
|---|---|
| H6.2 | The AV cannot utilize the maps. |
| H7 | The AV drives erratically on the road. [L 1,2,3,4,5,6,7,8,9] |
| H7.1 | The AV cannot follow the defined speed limit. |
| H7.2 | The AV accelerates and decelerates very abruptly. |
| H7.3 | The AV cuts across the defined lanes. |
| H7.4 | The AV stops abruptly |
| H8 | The AV is unable to switch between different modes of operation. [L 1,2,3] |
| H8.1 | The vehicle remains in autonomous mode and does not revert to manual mode |
| H8.2 | The vehicle remains in the manual mode despite initiating the autonomous mode. |

Following the identification of hazards, constraints on the system behaviour were estimated by inverting the hazard. These are listed out using the methodology described in section 2.1.1. The complete list of system-level constraints, Table B.1, has been added to the Appendix in Section B.1. Constraints identified for the two hazards are listed below:

1. The automated driving vehicle (AV) must maintain spacing between neighbouring vehicles based on defined requirements. Since the hazard has been broken down into two sub-hazards, the constraints derived from them are as follows:
   (a) **For the hazard H1.1:**
      i. The AV must maintain the required speed based on the speed limit data obtained from the HD maps.
      ii. The AV must follow a uniform acceleration profile to maintain the required levels of comfort for the passengers of the vehicle.
   (b) **For the Hazard H1.2:**
   The AV must initiate quick deceleration in the case of an emergency.
2. The hazard H2 has two system constraints placed on it, which are as follows:
   (a) The AV must engage autonomous mode only in regions/ road areas specified in the HD map.
   (b) In case of an error in mode detection, the AV must initiate a takeover request (TOR) immediately.

## 4.3 Detailed control structure diagram

Following the system level constraints, control structures of the system will be developed. Control is established from higher to lower blocks. Control and data flows from block to block. Human operators have also been incorporated within the given control loop. The identified control structures for the automated driving system have been described below.

### 4.3.1 High-Level (HL) control structure of automated driving vehicle using maps

Figure 4.1 describes the high-level architecture of the automated driving system. The vehicle under analysis is a SAE level 2/3 automated driving vehicle which utilizes high definition (HD) maps [17]. The highest level of control is established by the driver who is in control of the vehicle, which is manufactured by an OEM. The OEM defines a scope of the map based on which the mapmakers must develop a map. The automated driving vehicle (AV) is under the control of the

driver until the engagement of the autonomous mode. If the software system issues a take-over request, the driver must oblige and assume control of the vehicle. The AV uses HD map data to drive autonomously. Features of the map were read by the vehicle software system. HD maps were sourced from the environment, wherein features crucial for automated driving were filtered. Vehicle data is constantly uploaded to mapmakers, which were used for validating and updating existing maps.

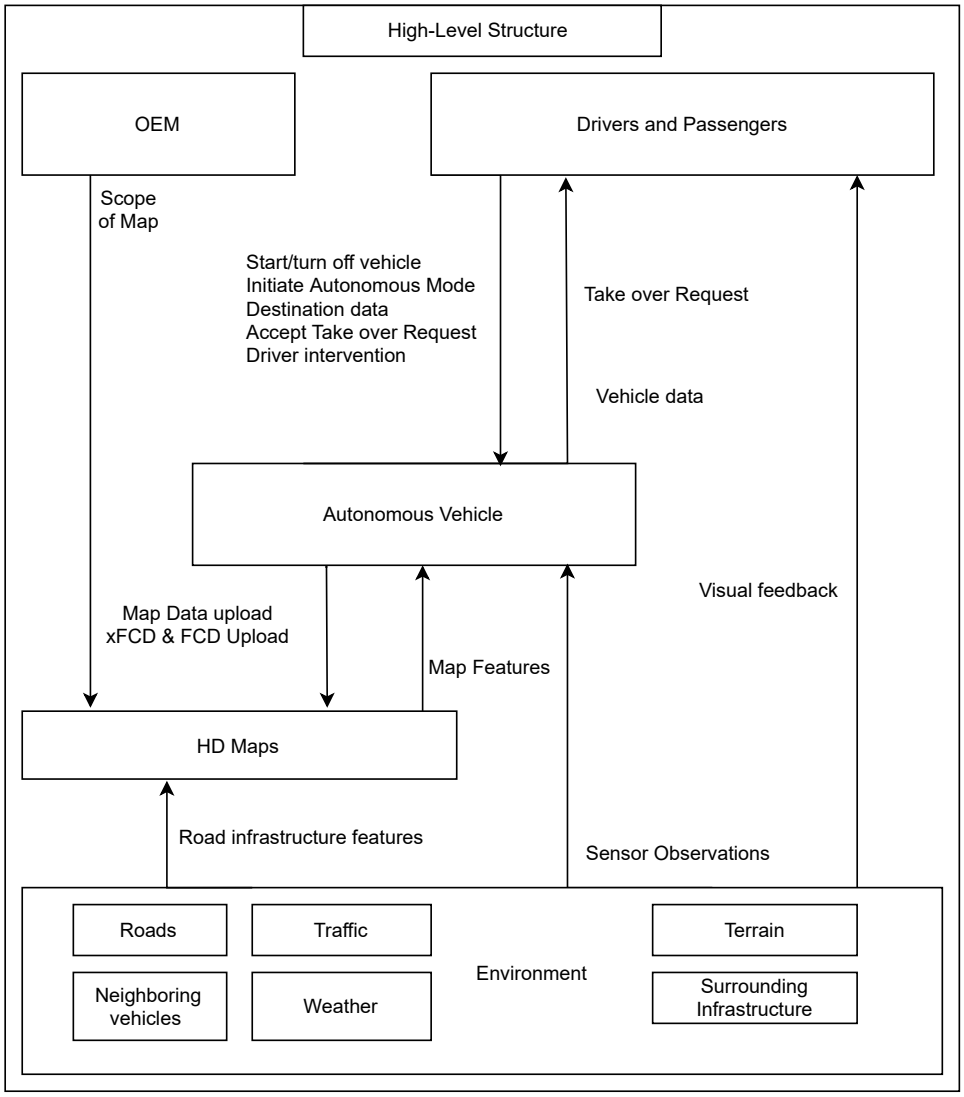


Figure 4.1: HL control structure of automated driving with maps.

### 4.3.2 Control structure of automated driving system

Figure 4.2 illustrates the software architecture of the vehicle. The vehicle HMI is the command center for the vehicle. The HMI is designed to aid the driver in establishing control of the vehicle. Furthermore, it displays vital data that the driver must observe to ensure the system is working smoothly. The inputs from the driver are communicated to the vehicle software system. The vehicle software system is responsible for executing commands from and providing feedback to the HMI. The vehicle software system receives input from sensors mounted on the vehicle. It receives HD map data from the map providers.

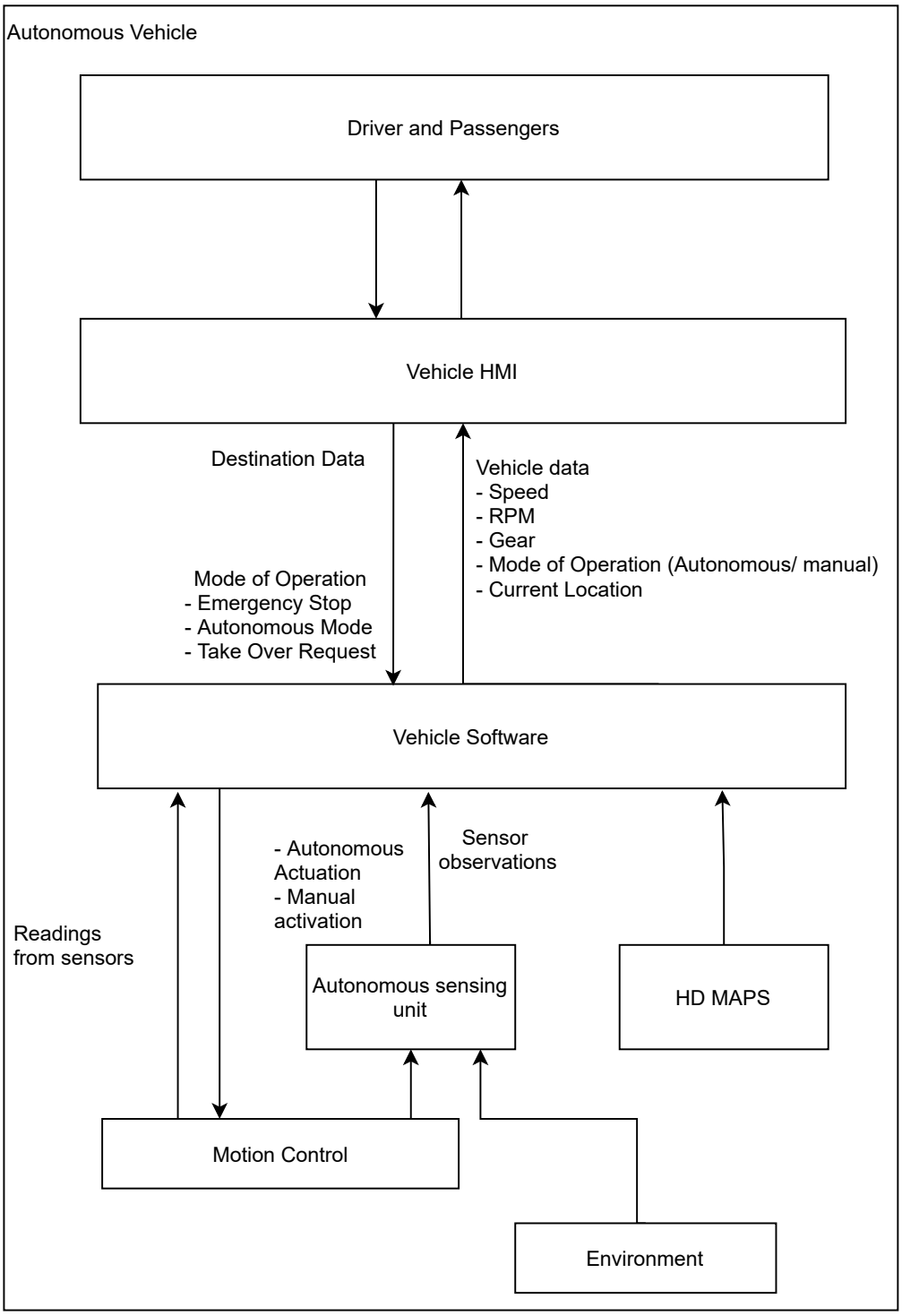


Figure 4.2: Control structure of automated driving system.

### 4.3.3 Low—Level (LL) Control structure for the vehicle software system

Figure 4.3 represents the low-level control structure of the vehicle software system. The software system were broadly split into 4 distinct layers. The topmost layer which consists of routing is responsible for receiving the output of and communicating data to the HMI. Based on the

input received from the driver, a route is generated. To follow a given route, the vehicle must generate a path. This is done by the path-planning subsystem. This constitutes the second layer of the system. In the third layer; perception, localization, and control are present. Perception is responsible for gathering data from the autonomous vehicle sensing unit. Localization utilizes the collected data from the perception unit and the HD map to estimate the vehicle's current pose. The current location of the vehicle is also estimated using this data. The control subsystem is responsible for engaging or disengaging the motion control unit of the vehicle which consists of the power unit, brakes, steering, and transmission. The final layer consists of motion control and AV sensing units. The AV sensing unit consists of sensors such as LiDAR, radar, camera, IMU, and GNSS receivers. This unit is crucial for deploying AV systems. Data is gathered from the environment and communicated to higher levels of control within the software system.

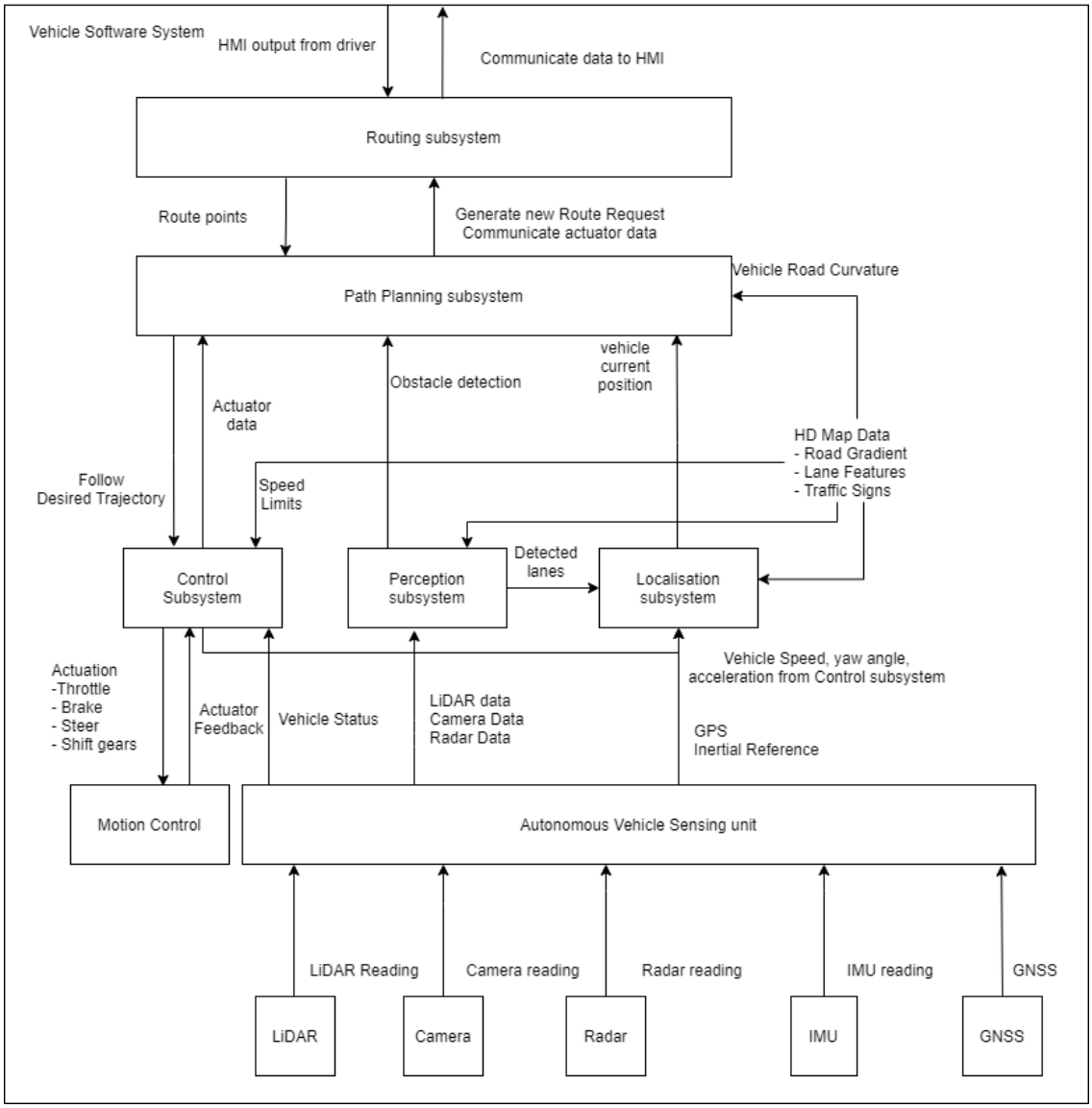


Figure 4.3: LL control structure of vehicle software system.

The remaining modelled control structures have been attached to the Appendix in Section B.2. Brief descriptions of each modelled control structure has also been provided.

## 4.4 Control actions, Unsafe control actions and controller constraints

Modelled control structures are used for listing out control actions, which are issued by different controllers in systems or sub-systems. Control actions are listed based on the origin of the control and the function which was being executed. Furthermore, the controller which receives the output of the control action has been highlighted. This ensures communication between different controllers in the system has been appropriately indicated and would aid in the further analysis of the same. In addition to the description of the control action, the table includes the details of the system/ sub-system which receives the output of this control action. This aids in tracing the impact of a control action on subsequent systems which are dependent on it. The type of dependency of each control action has been split into two categories: Map dependent and automated driving (AD) system dependent. A map dependent control action depends on the availability and quality of map data provided by the map delivery system. An automated driving (AD) system dependent function relies mainly on the output gathered from vehicle's sensors and controllers. In Tables 4.3 and 4.4, control actions performed by the perception algorithm are highlighted. The control structure from which the control actions below have been identified are observed in Figure B.2.

Table 4.3: Control action (CA): Detect lane markings and other road markings.

| | |
|---|---|
| Control action (CA) | Detect lane markings and other road markings |
| Controller responsible | Perception algorithm |
| System receiving output of control action | Path Planning Memory Unit |
| Description | The perception algorithm detects the lanes and lane markings on the road using map data and observational data received from the data storage unit (DSU). |
| Type of Dependency | Automated driving (AD) system and Map-Dependent |

Table 4.4: Control action (CA): Detect traffic signs.

| | |
|---|---|
| Control action | Detect traffic signs |
| Controller responsible | Perception algorithm |
| System receiving output of control action | Path Planning Memory Unit |
| Description | The perception algorithm detects the traffic signs using map data and observational data received from the data storage unit (DSU). |
| Type of Dependency | Automated driving (AD) system and Map-Dependent |

Unsafe control actions are listed out using the defined control actions as an input. The four categories mentioned in Section 2.1.3 were applied to each of the identified control actions. Table 4.5 highlights the results of applying the categories to the control action defined in Table 4.3. Each cell followed by the defined category is an unsafe control action which is performed by the controller. The resulting hazards caused due to the unsafe control actions have been identified at the end of each unsafe control action's description.

As mentioned in Section 2.1.3, unsafe control actions are inverted to form constraints which are then applied to the controller causing them. Table 4.6 highlights the constraints which have been identified for perception algorithm which performs the control action mentioned in Table 4.3. The identified constraints are enforced on the controller to prevent the occurrence of the identified unsafe control actions. Since the project aligned towards the identification of loss scenarios, the process of controller constraint identification was not given an impetus when compared to the remaining processes which have been conducted.

Table 4.5: Unsafe Control actions (UCA) for Detect lanes and other lane markings.

| Control action | Detect lanes and other lane markings |
|---|---|
| Unsafe Control action Categories | |
| Not Providing causes a hazard | 1. The perception algorithm cannot detect lane markings on the road during autonomous operation.[H 1,3,7] |
| | 2. The perception algorithm cannot detect lane crossings on the road during autonomous operation.[H 6,7] |
| Providing causes a hazard | 1. The perception algorithm detects incorrect lane positions during autonomous operation.[H 1,3,6,7] |
| | 2. The perception algorithm yields incomplete lanes during autonomous mode.[H 3,6] |
| | 3. The perception algorithm detects incorrect lane centerlines.[H 3] |
| Too early, too late or out of order | The perception algorithm detects lanes too late during autonomous operation.[H 1,6,7] |
| Stopped too soon or applied too long | NA |

Table 4.6: List of controller constraints (CC) for the defined unsafe control actions (UCA): Detect lanes and other lane markings.

| **UCA description** | **CC description** |
|---|---|
| The perception algorithm cannot detect lane markings on the road during autonomous operation.[H 1,3,7] | The perception algorithm must detect lane markings with the defined confidence during autonomous operation of the vehicle. |
| The perception algorithm cannot detect lane crossings on the road during autonomous operation.[H 6,7] | The perception algorithm must detect crossings with the defined confidence during the autonomous operation of the vehicle. |
| The perception algorithm detects incorrect lane positions during autonomous operation.[H 1,3,6,7] | The perception algorithm must detect lane markings with the defined confidence during the autonomous operation of the vehicle. |
| | If the perception algorithm is detecting lanes with a lower confidence level, the VSS must issue a TOR to the driver and must disable the autonomous mode (AM). |
| The perception algorithm yields incomplete lanes during autonomous mode.[H 3,6] | The perception algorithm must yield complete lane markings for the PP subsystem to validate and generate a new path. |
| The perception algorithm detects incorrect lane centerlines.[H 3] | The perception algorithm must use the lane centerlines in map data if the ASU cannot yield lane centerlines with the appropriate confidence. |
| The perception algorithm detects lanes too late during autonomous operation.[H 1,6,7] | The perception algorithm must detect lanes within the defined timing bounds. |

## 4.5 Loss scenarios and their categorisation

In the previous sub-sections, control actions, unsafe control actions and their respective controller constraints were identified for the automated driving system. Scenarios have been identified for the above defined unsafe control actions. This step has been conducted to list out possible situations wherein the system will provide the unsafe control action, which would lead to a hazard. The automated driving system receives map and its sensor information in the form of a data stream. Since the focus has been placed on data, the two categories which were used for listing of scenarios are as follows:

1. *Lack of information/feedback*
2. *Inadequate feedback/information is received*

Scenarios were listed out for the control actions having a map dependency. Control actions which had an automated driving system and map dependency were also considered whilst identifying loss scenarios. Scenarios identified for the control actions mentioned in Table 4.3, are presented in Table 4.7 below. The list of scenarios identified for the control action in Table 4.4 are presented in Table B.2, which has been attached to the Appendix in Section B.3.1.

In Figure 4.4, the second scenario from Table 4.7 is presented. The vehicle in automated driving mode (AM) approaches a road with missing lane markings. Due to the lack of information regarding lane markings on the road, the perception algorithm cannot detect lanes. The inability to provide lane information results in the localization system's failure to estimate the vehicle's lateral position with respect to the lane markings. As a result, the vehicle loses its lateral tracking momentarily. To track the path further, the vehicle must rely on the map data to drive to a location where the control can safely be handed over to the driver. If the vehicle receives inaccurate lane centerline information, the vehicle would cause a hazard such as H3, highlighted in Table 4.2.

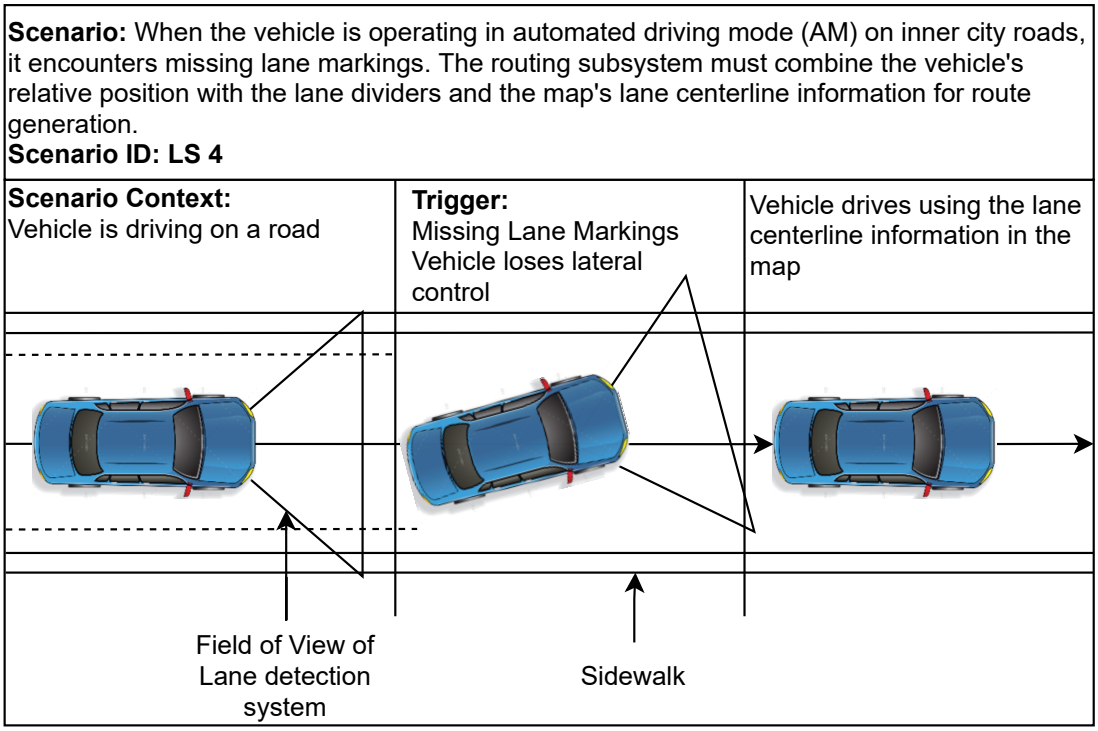


Figure 4.4: Loss scenario concerning missing lane markings on the road.

Table 4.7: List of scenarios for Control Action: Detect lanes and other lane markings.

| Unsafe Control action | Scenario classification | Scenario |
|---|---|---|
| The perception algorithm cannot detect lane markings on the road during autonomous operation.[H 1,3,7] | Inadequate information /feedback is received | When the vehicle is operating in autonomous mode in dimly lit conditions, the autonomous sensing unit (ASU) cannot view lane markings with the required confidence levels. Due to low confidence levels of observational data, the perception algorithm cannot detect lane markings. **Vehicle intent:** Detect and follow lanes on the road **Role of the map:** Provide lane borders, trajectories features for lane detection |
| | Lack of information | When the vehicle is operating in automated driving mode (AM), the road has missing lane markings. The autonomous sensing unit (ASU) cannot observe any lane markings, thus the perception algorithm cannot detect lane markings and must rely on the road features provided within the map. **Vehicle intent:** Detect and follow lanes on the road **Role of the map:** Provide required map features for lane detection |
| The perception algorithm cannot detect lane crossings on the road during autonomous operation.[H 6,7] | Inadequate information /feedback is received | When the vehicle is driving in automated driving mode (AM) on an inner-city road, it contains faint/old markings at lane crossings. The perception algorithm cannot detect incomplete/old markings on the road thus does not stop at any crossing in its defined path. **Vehicle intent:** Detect and follow lanes on the road **Role of the map:** Provide required map features for lane detection |
| | Lack of information | When the vehicle is operating in automated driving mode (AM) during heavy rain, it drives through an underpass where the roads are covered/flooded with water. The flooding covers the lane markings completely, which cannot be identified by the ASU. **Vehicle intent:** Detect and follow lanes on the road **Role of the map:** Provide lane borders features for lane detection |
| The perception algorithm detects incorrect lane positions during autonomous operation.[H 1,3,6,7] | Inadequate information /feedback is received | When the vehicle is in automated driving mode (AM), the perception algorithm uses obscured camera data (due to foggy, rainy, snowy conditions) for the detection of lanes with respect to lane features in the map. **Vehicle intent:** Detect and follow lanes on the road **Role of the map:** Provide lane features required for lane detection |
| | Lack of information | When the vehicle is driving in automated driving mode (AM), the camera encounters lane markings which are difficult to process (yellow-colored lane markings) thus forcing the routing subsystem to use line features from the map data for route generation. **Vehicle intent:** Detect and follow lanes on the road **Role of the map:** Provide lane features required for lane detection |
| The perception algorithm yields incomplete lanes during autonomous mode.[H 3,6] | Inadequate feedback /information is received | When the vehicle is in automated driving mode (AM), due to inadequate street lighting, the ASU can observe the lane markings only in instances when sufficient light is available. This results in incomplete lane markings being identified by the perception algorithm. **Vehicle intent:** Detect and follow lanes on the road **Role of the map:** Provide required map features for lane detection |
| The perception algorithm detects incorrect lane centerlines.[H 3] | Inadequate feedback /information is received | When the vehicle is in automated driving mode (AM), the map delivery system provides inaccurate lane features to the perception algorithm for the detection of the lane centerlines. This results in incorrect lane centerlines being followed by the vehicle. **Vehicle intent:** Detect and follow lanes on the road Role of map: Provide required map features for lane detection |
| The perception algorithm detects lanes too late during autonomous operation.[H 1,6,7] | Inadequate feedback /information is received | When the vehicle is in automated driving mode (AM), the ASU provides lane observation data later than expected due to environmental conditions such as low lighting and rainy conditions. **Vehicle intent:** Detect and follow lanes on the road **Role of the map:** Provide required map features for lane detection |
| | Inadequate feedback /information is received | When the vehicle is in automated driving mode (AM), the vehicle navigation system (VNS) requires longer processing time for reading the road features required for lane detection by the perception algorithm. **Vehicle intent:** Detect and follow lanes on the road **Role of map:** Provide required map features for lane detection |

Figure 4.5 illustrates the loss scenario, LS 5, which occurs due to incorrect speed limit information retrieved from the map. The blue and red vehicles are driving in the outer and inner lanes respectively. The red vehicle blocks the blue vehicle from detecting the speed limit sign, which is a 30 kph restriction. The blue vehicle retrieves map information regarding the missed speed limit, which is inaccurate. The blue vehicle begins speeding up to ensure it meets the required speed limit as per the map data, which results in it outpacing the red vehicle. In this scenario, the blue vehicle could encounter a hazard, H1, presented in Table 4.2.

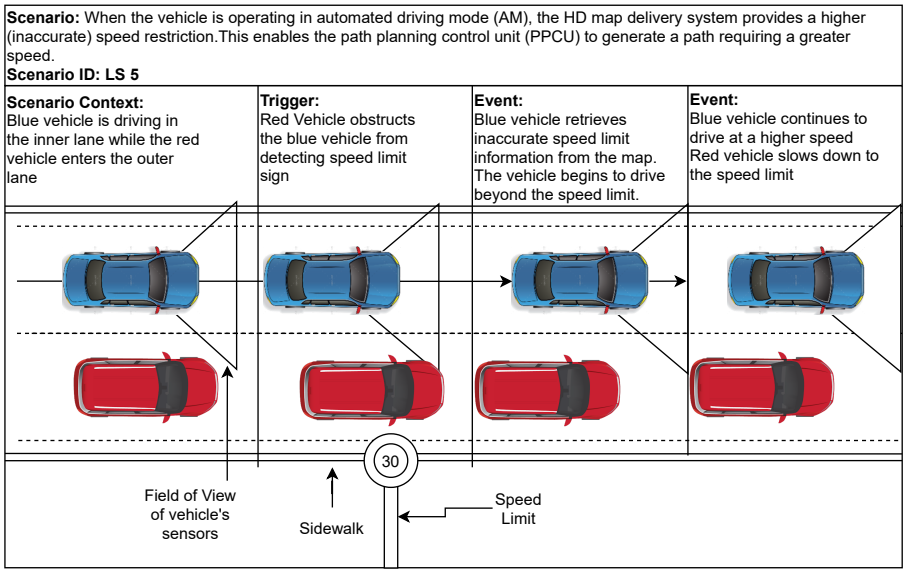


Figure 4.5: Loss scenario resulting from incorrect speed limit information received from map.

Figure 4.6 illustrates the loss scenario, LS 6, which occurs due to incorrect road curvature information in the map. The vehicle is driving in AD mode and is about to encounter a road with a defined radius of curvature. There is no traffic sign indicating the magnitude of the incoming curvature. Thus, the vehicle utilizes the required radius of curvature information from the map. However, the data received from the map is larger than the expected radius. The vehicle begins to manoeuvre a path which takes the vehicle close to the lane borders. This causes the vehicle to encounter a hazard, H7 and H3, presented in Table 4.2.

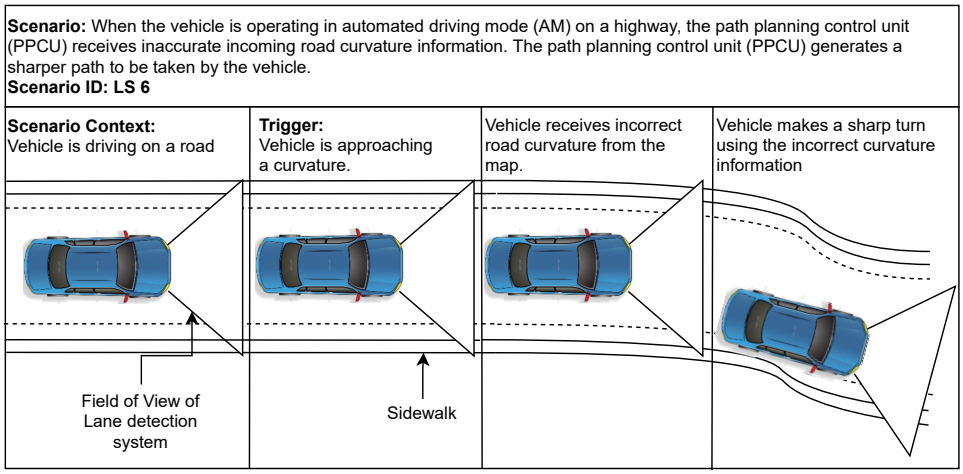


Figure 4.6: Loss scenario resulting from incorrectly received road curvature from map.

Figure 4.7 illustrates the loss scenario, LS 14. The blue vehicle is operating in AD mode in the outer lane and the inner lane is occupied by the red vehicle. The stop sign is placed at a blind spot, which goes undetected for the blue vehicle. The blue vehicle continues to operate at its selected speed limit without being aware of the incoming stop line. The red vehicle begins to slow down as it approaches the stop line. The blue vehicle utilises the map data to retrieve information about incoming or past traffic signs. It receives the stop sign information and also detects the incoming stop line. The blue vehicle engages the brakes, therefore narrowly comes to a halt at the defined stop line.

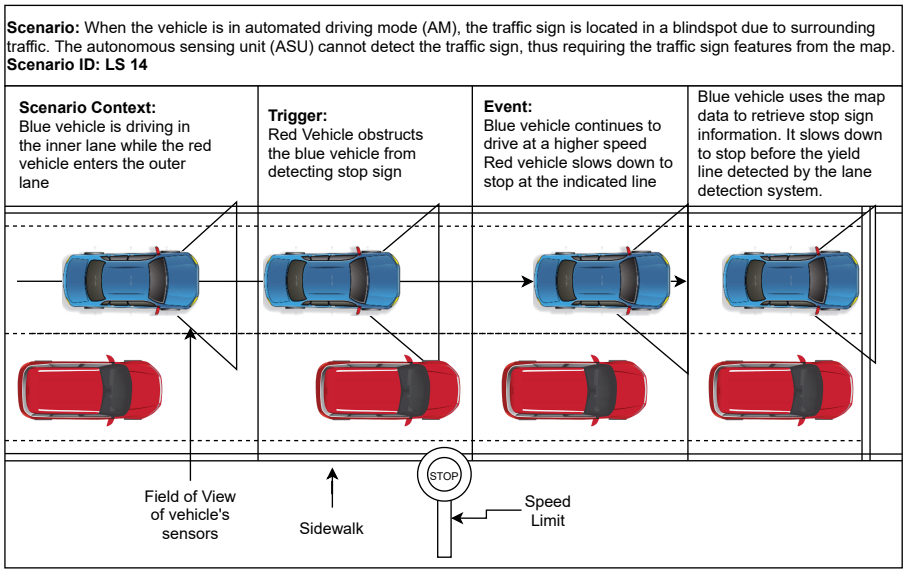


Figure 4.7: Loss scenario resulting from blocking of camera.

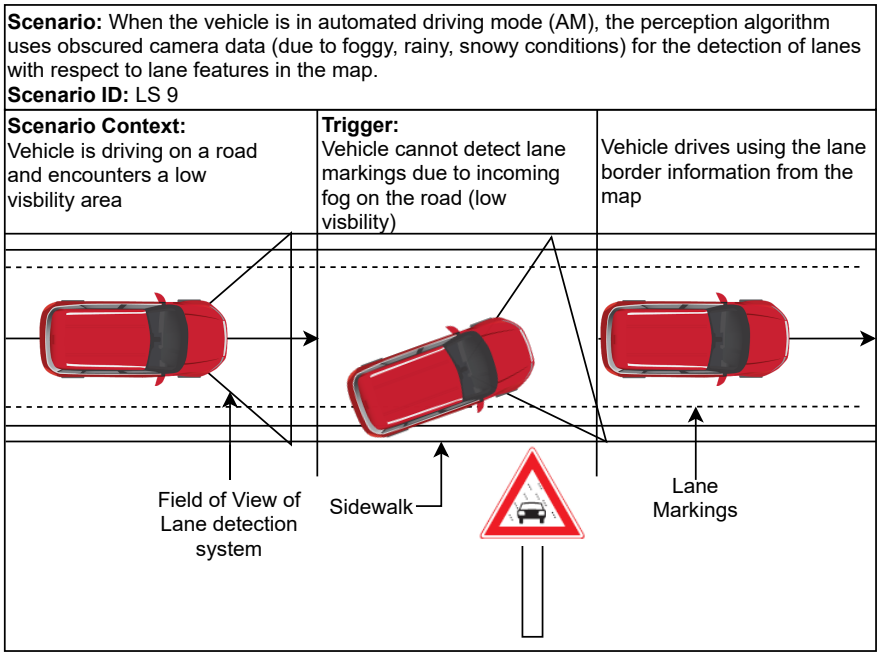


Figure 4.8: Loss scenario resulting from low visibility caused by foggy weather.

Figure 4.8 illustrates the loss scenario, LS 9. The vehicle is operating in AD mode during which it approaches a foggy area. The vehicle enters the low visibility area and cannot detect lane markings with the required confidence. This leads to a drop in the performance of the lane-keeping of the vehicle. The vehicle must retrieve lane border information from the map to ensure it stays within the defined lanes for the entirety of the given fog covered area.

### 4.5.1 Categorisation of loss scenarios:

As mentioned in Section 2.1.4, scenarios are categorised based on their severity and probability of exposure. Each scenario in the pool of scenarios was assigned a value of severity and exposure. The highest level of severity and probability of exposure is 3. The product of the severity and exposure was used to estimate the priority of a given scenario. A scenario is considered to be highly probable and most severe if the product of severity and probability of exposure is 9 and the opposite holds true. Table 4.8 presents the level of priority for different products of exposure and severity. A short list of high-priority scenarios is presented in Table 4.9.

Table 4.8: Levels of priority based on product of severity and probability of exposure.

| **Level of Priority** | **Product of probability of exposure and severity** |
|---|---|
| Low Priority | 1 |
| Medium Priority | 2,3,4,6 |
| Highest Priority | 9 |

Table 4.9: List of high-priority scenarios.

| **Scenario ID** | **Scenario Description** |
|---|---|
| LS 4 | When the vehicle is operating in automated driving mode (AM) on inner city roads, it encounters missing lane markings. The routing subsystem must combine the vehicle's relative position with the lane dividers and the map's lane centerline information for route generation.<br>**Vehicle intent:** Use map data for generating a route<br>**Role of map:** Provide required map data needed for generation of a route<br>**KPI of features:** Positional (Absolute and relative) accuracy, completeness of lane borders, trajectories |
| LS 5 | When the vehicle is operating in automated driving mode (AM), the HD map delivery system provides a higher (inaccurate) speed restriction. This enables the Path planning Control unit (PPCU) to generate a path requiring a greater speed.<br>**Vehicle intent:** generate a path using the selected route<br>**Role of map:** Provide speed restrictions features for path generation<br>**KPI of features:** Thematic accuracy and completeness (false positive and negatives) of speed restrictions |
| LS 6 | When the vehicle is operating in automated driving mode (AM) on a highway, the Path planning Control unit (PPCU) receives inaccurate incoming road curvature information. The PPCU generates a sharper path to be taken by the vehicle.<br>**Vehicle intent:** generate a path using the selected route<br>**Role of map:** Provide point features (incoming road curvature) for path generation<br>**KPI of features:** Heading accuracy and absolute accuracy of curvature |

Each of the above-mentioned scenarios are presented in Figures 4.4, 4.5, and 4.6. The complete list of high-priority scenarios is attached in B.3. The breakdown of scenarios with respect to

features in the map has been presented in Table 4.10. Lane features and traffic features have the most high-priority scenarios in comparison with the remaining features of the map.

Table 4.10: Number of scenarios for each feature present in the map.

| **Map feature** | **Number of high-priority scenarios** |
|---|---|
| Lane Features | 7 |
| Traffic features (signs and lights) | 7 |
| Speed restrictions | 1 |
| Overhead structures | 2 |

### 4.5.2 Root-cause analysis of high-priority loss scenarios

Following the classification process, a root-case analysis was performed on the list of scenarios. The analysis was performed to understand the type of scenario, different entities present, and the events occurring in the given scenario [50]. The results of the analysis performed for scenarios highlighted in the previous section are presented in Table 4.11. The complete list of results are presented in Table B.4, which is placed in the Appendix in Section B.3.3.

Table 4.11: Root-cause analysis of high-priority scenarios (LS4, LS5 and LS 6).

| **Loss scenario ID** | **Unforesee -able** | **Foresee-able** | **Dynamic Entities** | **Static Entities** | **Events** |
|---|---|---|---|---|---|
| LS 4 | NA | Unpreventable | Road features (lane markings) | Vehicle, traffic | Missing lane markings → failure to identify lanes |
| LS 5 | NA | Preventable | Speed limit signs, Road features | Vehicle, traffic | Inability to read speed limit signs → Incorrect speed limit received from map |
| LS 6 | NA | Unpreventable | Road features, (curvature signs) | Vehicle, traffic | Incorrect road curvature information received from map |

Loss scenarios, LS4 and LS6, are classified as unpreventable scenarios whereas LS5 has been classified as a preventable scenario. LS 4 is an unpreventable scenario since it occurs when the vehicle drives on a road with missing lane markings. This can occur at any given moment of time when the vehicle is in motion. Similarly, LS 6 occurs if inaccurate curvature information is provided to the vehicle from the map. Since the map data is provided to the vehicle directly, this situation is termed as unpreventable. Furthermore, this could be considered as a preventable scenario only if daily updates concerning the curvature are provided to the map .This would aid in preventing the occurrence of the scenarios.

Static entities refer to those actors which remain in their current positions in the scenario [50]. The static entities in the scenarios are the features of the road such as lane markings, traffic signs such as speed limit and incoming road curvature signs. Dynamic entities refer to those actors in a scenario which experience a state change [50]. The dynamic entities in the scenarios are the automated driving vehicle and traffic which includes the vehicles driving in the neighbouring lanes and pedestrians.

The event or trigger in a scenario is the cause for the scenario or what initiates the scenario. LS 4 is initiated when no lane markings are present on the road. This causes a failure of the lane

detection system due to which the vehicle must use lane centerline information to negotiate the given area. Similarly, LS 6 is triggered by receiving inaccurate road curvature information from the map. LS 5 is triggered by the inability to observe traffic signs. The vehicle utilises map data to retrieve the speed limit information, however the information received is inaccurate. In a similar manner, the remaining scenarios present in the list have been analysed.

## 4.6 Loss scenario validation

STPA yielded a list of high-priority loss scenarios in which an unsafe control action may be executed, resulting in the occurrence of a hazard. The analysis was performed to understand the usage of map from the point of view of an automated driving system. Scenario validation was performed to ensure that the loss scenarios are relevant and have been appropriately identified.

The process of validation begun by pooling the scenarios obtained from STPA performed, TomTom's client data, and TomTom's measurement data. STPA yielded a list of 17 loss scenarios, obtained after performing the categorisation. Table B.3 has been attached to the Appendix in Section B.3.2. The list of scenarios identified from TomTom's clients are presented in Table B.5, which has been attached to the Appendix in Section B.4.2. The list of scenarios identified using TomTom's measurement data are presented in Table B.6, which is attached to the Appendix in Section B.4.3.

The list of scenarios validated using the sources of data are presented in Table 4.12. The ID of each validated scenario, the source of the scenario, and the number of scenarios validated using that given source is mentioned in the table below.

Table 4.12: List of validated scenarios using sources of data.

| **Source of scenarios** | **ID of Validated scenarios (LS)** | **Number of validated scenarios** |
|---|---|---|
| TT Client data | 5,6,7,9,10,13,15,17 | 8 |
| TT MoMa data | 1,2,5,8,9,11,12,14,15,16 | 10 |
| TT client + MoMa data | 5,9,15 | 3 |

Figure 4.9 presents a Venn chart of the scenarios which have been validated using the two sources of data. Each source of data has validated almost half the scenarios identified from STPA. A total of 15 scenarios have been validated using either TomTom's client data or MoMa data, which accounts for more than 80% of the scenarios identified from STPA.

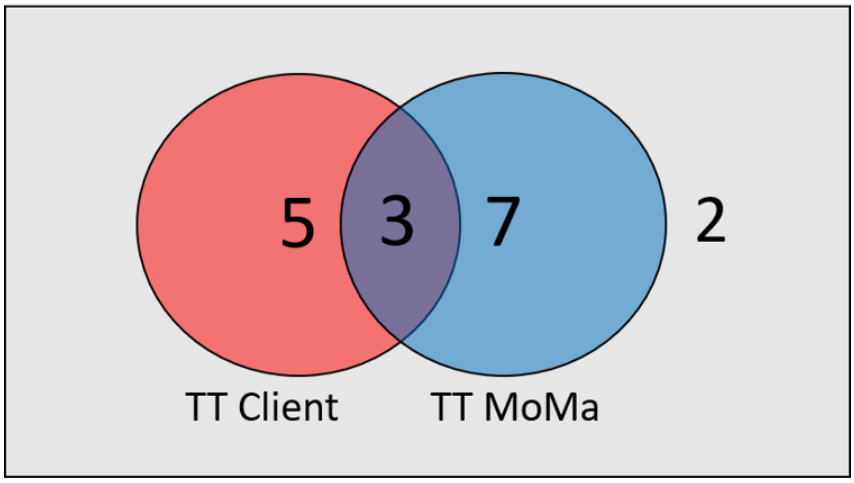


Figure 4.9: Venn diagram for validated scenarios.

Table 4.13 presents the validation of scenario LS 5 using TomTom client's data. STPA yields a scenario in which the vehicle receives inaccurate speed restriction information from the map. This results in the vehicle generating a path requiring a higher speed, which makes it an unsafe maneuver. The scenario identified by TomTom's client revolves around receiving inaccurate speed limit data from the map. Since both scenarios identify the same aspect about the map feature, speed limit data, LS 5 is considered as validated.

Table 4.13: Scenario validated using TomTom (TT) Client information.

| **Source of scenario** | **Scenario Description** |
|---|---|
| STPA | When the vehicle is operating in automated driving mode (AM), the HD map delivery system provides a higher (inaccurate) speed restriction. This enables the Path Planning Control unit (PPCU) to generate a path requiring a greater speed. |
| TT Client data | The vehicle receives an incorrect speed limit from the map data. |

Table 4.14 presents the validation of scenario LS 13 using TomTom Mobile Mapping (MoMa) data. The scenario describes the failure to detect traffic signs when there is excessive glare on a bright sunny day. Due to the excessive brightness, the camera cannot provide observational data required for sign detection. The AD system must rely on traffic sign information provided in the map. The scenario identified using MoMa data also presents the failure to detect traffic signs on a sunny day. Since both the sets of scenarios revolve around the failure to detect traffic signs in a given environmental setting, scenario LS 13 is considered as validated.

Table 4.14: Scenario validated using TomTom (TT) Mobile Mapping Vehicle (MoMa) data.

| **Source of scenario** | **Scenario Description** |
|---|---|
| STPA | When the vehicle is operated in automated driving mode (AM), the cameras are unable to capture the traffic signs due to excessive glare (bright sunny day). The perception algorithm must rely on traffic sign features provided in the map |
| TT MoMa data | The vehicle cannot detect traffic signs under sunny conditions. |

Table 4.15: Scenario validated using TomTom Mobile Mapping Vehicle (MoMa) and Client data.

| **Source of scenario** | **Scenario Description** |
|---|---|
| STPA | When the vehicle is operating in automated driving mode (AM) during heavy rain, it drives through an underpass where the roads are covered/flooded with water. The flooding covers the lane markings completely, which cannot be identified by the autonomous sensing unit (ASU). |
| TT MoMa data | The vehicle is unable to detect lanes under conditions of heavy rain |
| TT Client Data | The vehicle approaches an underpass which is flooded due to incessant rainfall. |

Table 4.15 presents the validation of scenario LS 8 using both TomTom's client and MoMa data. STPA identifies a scenario in which the vehicle cannot identify lane markings whilst driving through an underpass due to incessant rainfall. The vehicle's AD system utilises the lane features in the map to navigate through the underpass. TomTom's client partially validates the scenario by

identifying the same operating condition as mentioned in STPA. TomTom's MoMa data identifies the failure of lane detection when driving in heavy rain. Thus, both scenarios cumulatively validate LS 8.

The completion of the validation process concludes the safety analysis. The focus of the analysis was on identifying unsafe scenarios for the AD system. This led to the identification of 205 scenarios. These were further categorised to yield 17 high-priority scenarios. Fifteen scenarios were validated using either TomTom's client or measurement data.

# Chapter 5

# Results of CARLA simulator

In this chapter, the results obtained from the tests conducted in CARLA simulator are presented. The list of parameters which have been varied in each test are discussed. Observations drawn from each plot are presented in this chapter.

Using the list of high priority scenarios, the feature of the map which was deemed safety-critical for the vehicle was the lane features. The lane features of the map consist of the lane centerlines and borders. Furthermore, CARLA provided coordinates for navigation in the form of waypoints. In Section 3.3, a comparison was drawn between waypoints in CARLA and lane centerlines on the road. We have utilised the availability of lane centerline information in CARLA for conducting simulations. Thus, lane centerlines was the focus of the tests conducted in CARLA. The parameters which were taken into consideration whilst performing simulations in CARLA are presented in Table 5.1.

Table 5.1: List of parameters considered in simulations.

| **Parameters** | **Range of values** |
|---|---|
| Type of Noise | Gaussian |
| Bias (Mean Noise) | [-0.60, 0.60] meters |
| Jitter (Standard Deviation) | [0,0.20] meters |
| Curvature of road | [0,100] meters |
| Speed of vehicle | 28-30 kmph |
| Vehicle models | Prius, Cybertruck, Model 3 |
| Sampling size | [0.15, 2.5] meters |

The range of bias and jitter injected in the waypoints are varied from the Table 5.1. The range of bias and jitter was selected based on the quality levels set by TomTom for their products. A gaussian noise signal with the required bias and jitter was generated. The route for the vehicle has been generated using the method proposed in Section 2.3.3. Following the creation of the route, the noise was injected into the list, element by element. Therefore, each element in the list deviates from its original route. The vehicle attempts to follow the noise injected path, and the key performance indicators (KPIs) are measured and stored for each case. Before conducting the simulation, scenarios were selected for the vehicle. The first scenario taken under consideration is the vehicle is driving on a straight road. The second scenario is the vehicle manoeuvres a path with a defined curvature. The radius of curvature was varied within the range set in Table 5.1. The range of curvature was selected based on the maximum radii of roads available in CARLA's pre-defined maps. The results obtained by varying the set parameters has been presented in the upcoming sections.

## 5.1 Variation of sampling size of map

In this section, the sampling size or accuracy of the map generated in CARLA was varied. The sampling size is used as an input for generating a route for the vehicle [40]. The sampling size was varied in steps of 0.5 meters at higher magnitudes, followed by a variation of 0.25 meters at lower magnitudes of sampling size. The parameters considered for simulation are listed in Table 5.2. The vehicle of selection was driven on a straight road with the accuracy of the map being varied. The KPI, mean absolute error, was measured and is presented below in Figure 5.1.

Table 5.2: Parameters used in variation of accuracy.

| **Parameter** | **Value** |
|---|---|
| Vehicle | Prius |
| Noise | Nil |
| Vehicle speed | 30 kph |
| Map accuracy | [0,2.5] m |

In Figure 5.1, we can observe the decrease in the mean absolute error with a decrease in the sampling size of the map. The decrease in mean absolute error is much larger when the sampling size of the map is decreased from 2.5 to 2 meters as compared to a decrease from 1.5 to 1 meter. The rate of decrease in mean absolute error is lower when the step size approaches a value of 0.25 meters. When the sampling size is decreased below 0.25 meters, the mean absolute error stabilises.

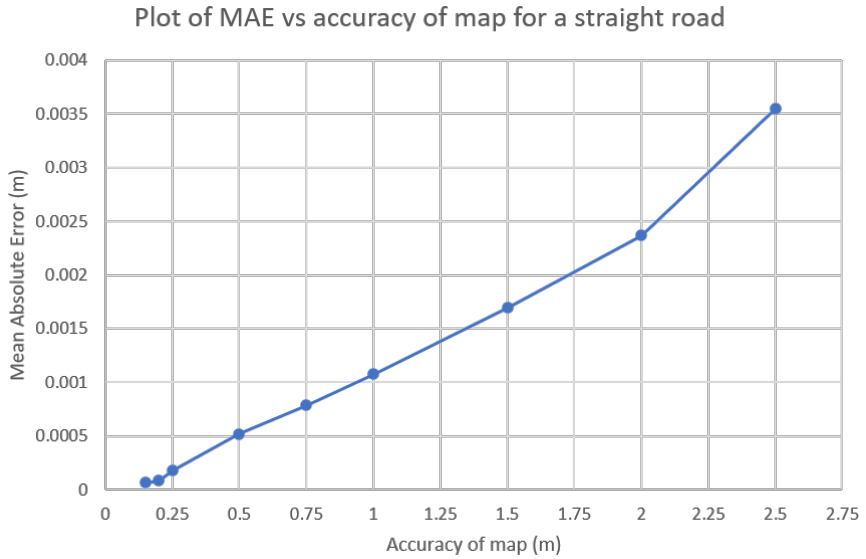


Figure 5.1: Plot of Mean absolute error vs Accuracy/ Sampling size of map on a straight road.

## 5.2 Variation of bias and jitter in lane waypoints on a straight road

The route taken by the vehicle is presented in Figure 5.2. Lateral noise was injected in the route. The speed of the vehicle was set to 30 kph and the vehicle model is Toyota Prius. A bias of the range -0.60 to 0.60 meters was coupled with an element of jitter ranging from 0.05 to 0.20 meters, to generate the required noise signal. The noise signal was injected in the vehicle's route and the data concerning KPIs was recorded.

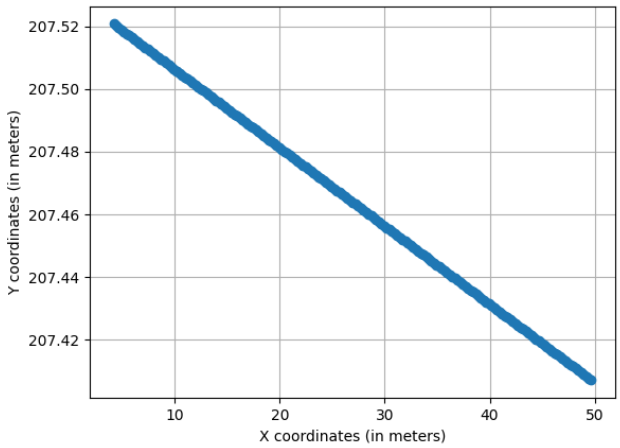


Figure 5.2: Route generated for the first scenario.

The results obtained for the cases having a jitter of 0.05, 0.10, 0.15, and 0.20 meters are presented in Figure 5.3 and Figure 5.4. The mean absolute error increases with increasing bias injected in the map. The mean absolute error peaks when the bias is set to 0.60 or -0.60 meters. In Figure 5.4, we observed higher number of lane invasions at higher magnitudes of bias injected in the lane centerlines. Higher lane invasions is observed in cases in which low values of bias are coupled with high values of jitter.

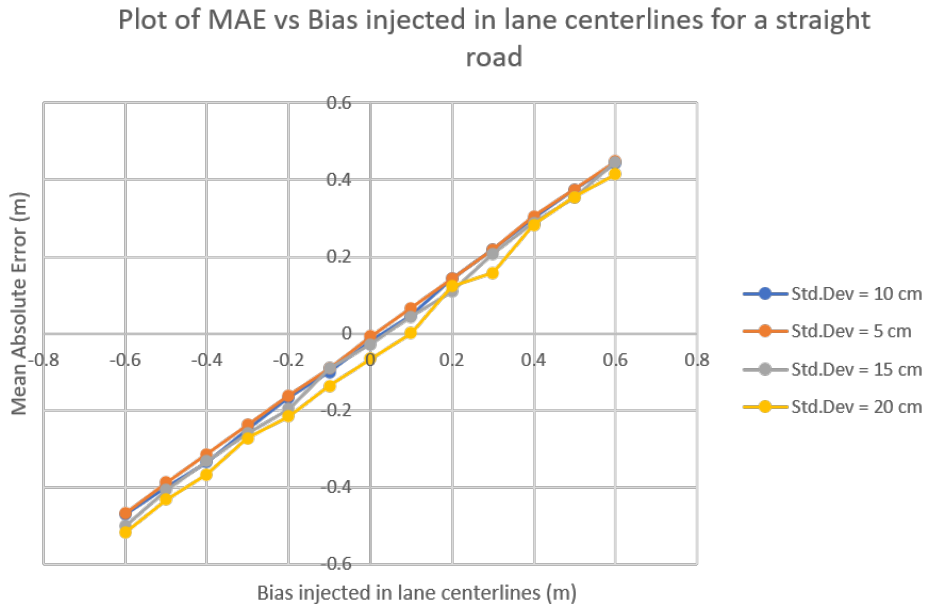


Figure 5.3: Plot of Mean Absolute error (MAE) vs Bias injected in lane centerlines (m) for a straight road.

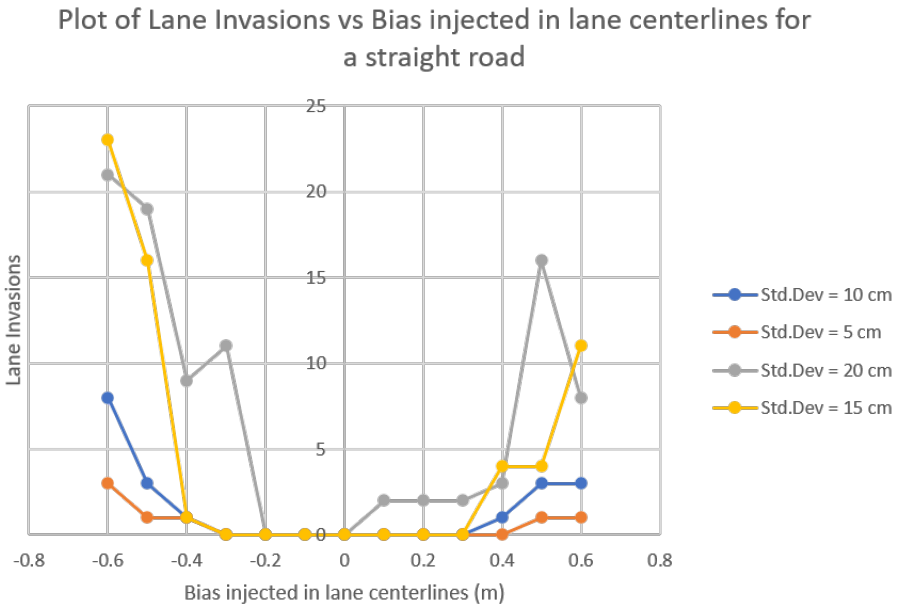


Figure 5.4: Plot of Lane invasions vs Bias injected in lane centerlines (m) for a straight road.

## 5.3 Variation of bias and jitter in lane waypoints on roads with curvature

In this sub-section, the second scenario has been tackled. The vehicle was driven on road with a defined curvatures. The route was created for the vehicle by adding a straight road which then leads up to the curvature. The vehicle drives straight forward and makes a left turn on the road with the set curvature. The curvatures are selected from the range provided in Table 5.1.

1. **Radius of curvature = 6.5 m**

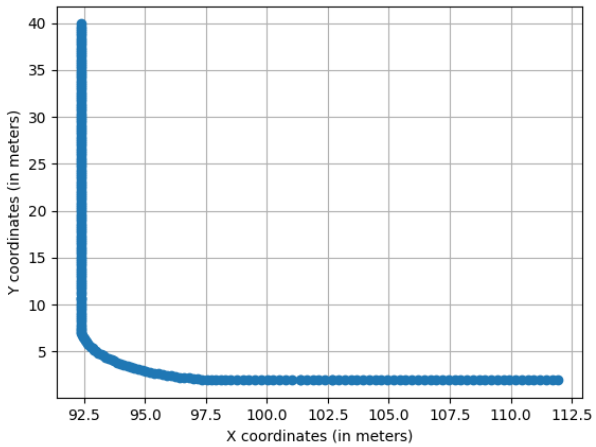


Figure 5.5: Route generated for road with radius of curvature = 6.5 m.

The vehicle manoeuvred a road with a radius of curvature equal to 6.5 meters. This radius

was selected to replicate a sharp inner city turn. The route followed by the vehicle is presented in Figure 5.5. The speed of the vehicle was set to 30 kph and the vehicle model was Toyota Prius.

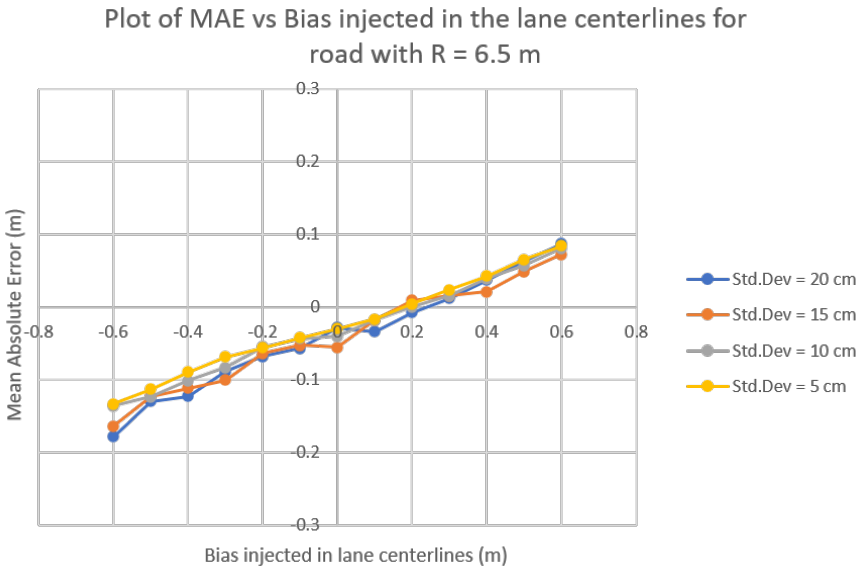


Figure 5.6: Plot of Mean Absolute error (MAE) vs Bias injected in lane centerlines for road with 6.5 meters curvature.

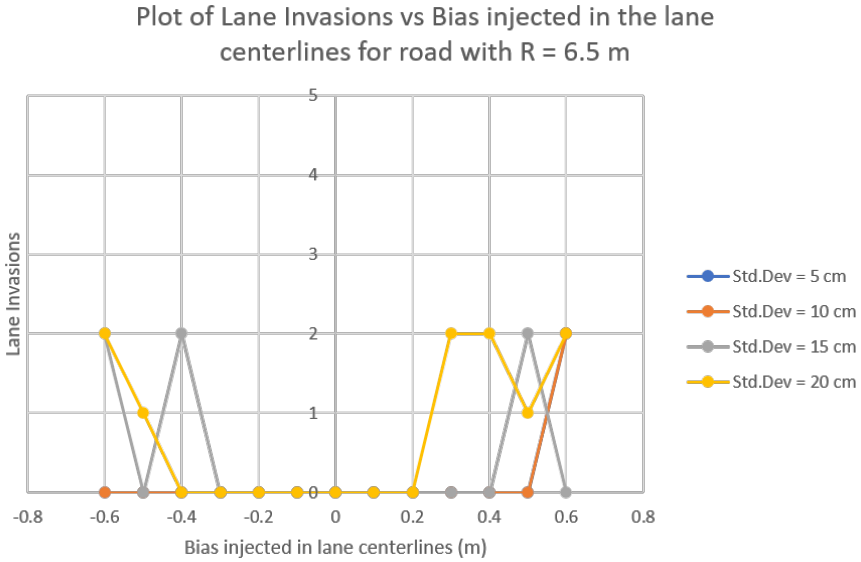


Figure 5.7: Plot of Lane Invasions vs Bias injected in lane centerlines for road with 6.5 meters curvature.

In Figure 5.6, we observed the increase in mean absolute error when the magnitude of bias injected in the lane centerlines was increased. When the bias is increased from 0 to 0.60 metres, the impact of varying jitter was minimal. When the bias is increased from -0.60 to 0 metres, the mean absolute error increases with increasing jitter. The mean absolute

error peaked at a bias of 0.60 metres. In Figure 5.7, the number of lane invasions increased with increasing jitter. However, the maximum number of lane invasions was limited to two despite increasing jitter in the noise signal.

2. **Radius of curvature = 62 m**

   The vehicle manoeuvred a road with a radius of curvature equal to 62 meters. The route followed by the vehicle is presented in Figure 5.8. The speed of the vehicle was set to 30 kph and the vehicle model was Toyota Prius.

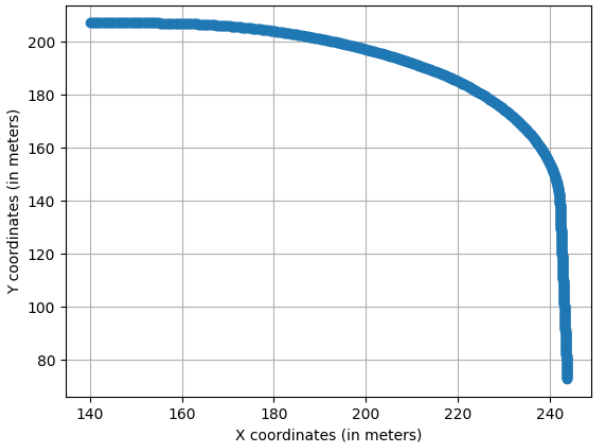


Figure 5.8: Route generated for road with radius of curvature = 62 m.

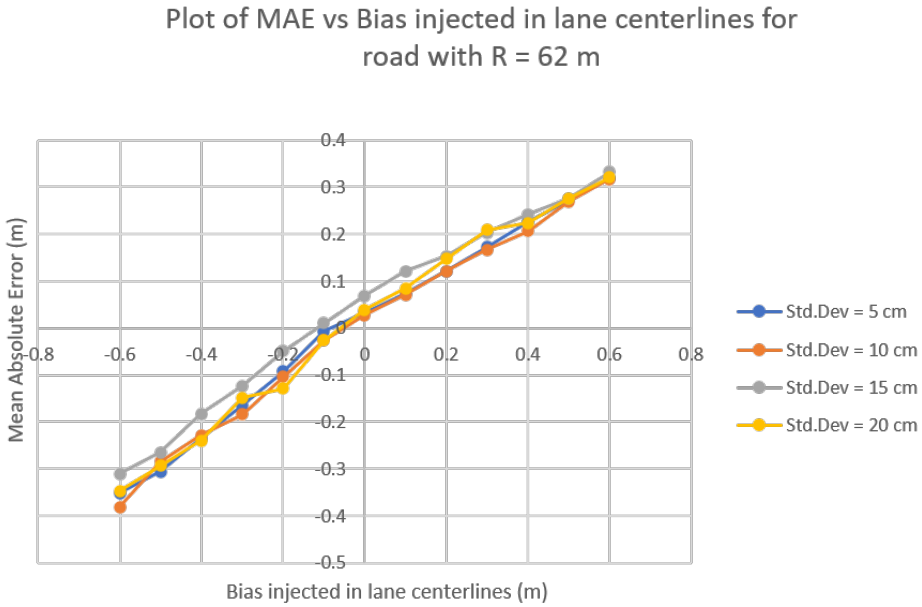


Figure 5.9: Plot of Mean Absolute error (MAE) vs Bias injected in lane centerlines for road with 62 meters curvature.

The results obtained for the cases having a jitter of 0.05, 0.10, 0.15, and 0.20 meters are presented in Figure 5.9 and Figure 5.10. The mean absolute error increased with increasing

bias injected in the map. The mean absolute error peaked when the bias was set to 0.60 or -0.60 meters. The mean absolute error peaked to higher value at -0.60 meters of bias when compared to 0.60 meters. In Figure 5.3, we observed the increase in the peak of mean absolute error with increasing standard deviation or jitter.

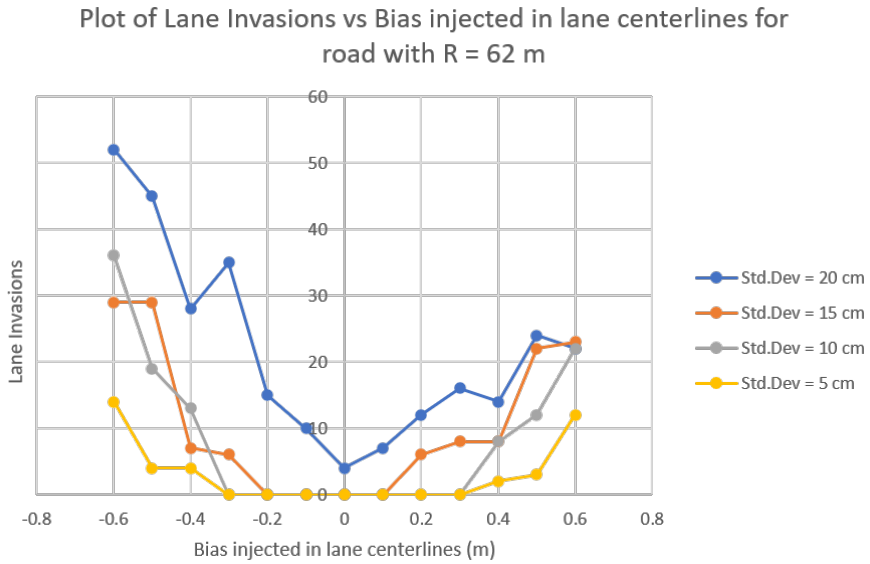


Figure 5.10: Plot of Lane invasions vs Bias injected in lane centerlines for road with 62 meters curvature (m).

In Figure 5.10, maximum lane invasions are observed for a bias and jitter of -0.60 and 0.20 centimeters, respectively. The number of lane invasions increased when the magnitude of jitter was raised from 0.05 to 0.20 meters. In Figures 5.10 and 5.4, a vertical parabolic shape (U-shaped) was observed. The vertical parabolic shape moved towards the Y axis,i.e lane invasions, with an increase in jitter, which is observed in Figure 5.10.

3. **Radius of curvature = 102 m**

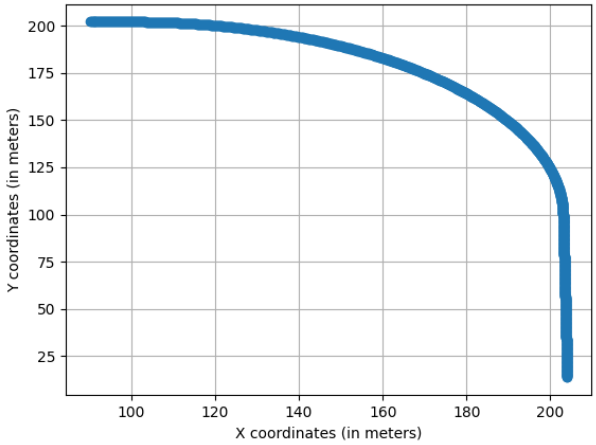


Figure 5.11: Route generated for road with radius of curvature = 102 m.

The vehicle manoeuvred a route with a radius of curvature equal to 102 meters. The route followed by the vehicle is presented in Figure 5.11. The curvature was selected to emulate a scenario where the vehicle drives on a highway where the radius of curvature of the roads are large. The speed of the vehicle was set to 30 kph and the vehicle model was Toyota Prius.

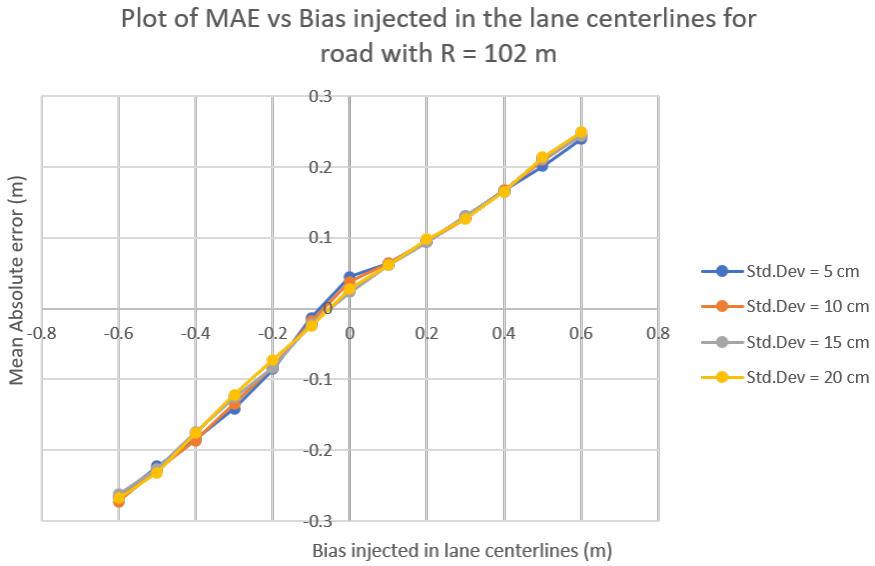


Figure 5.12: Plot of Mean Absolute error vs Bias injected in lane centerlines for road with 102 meters curvature (m).

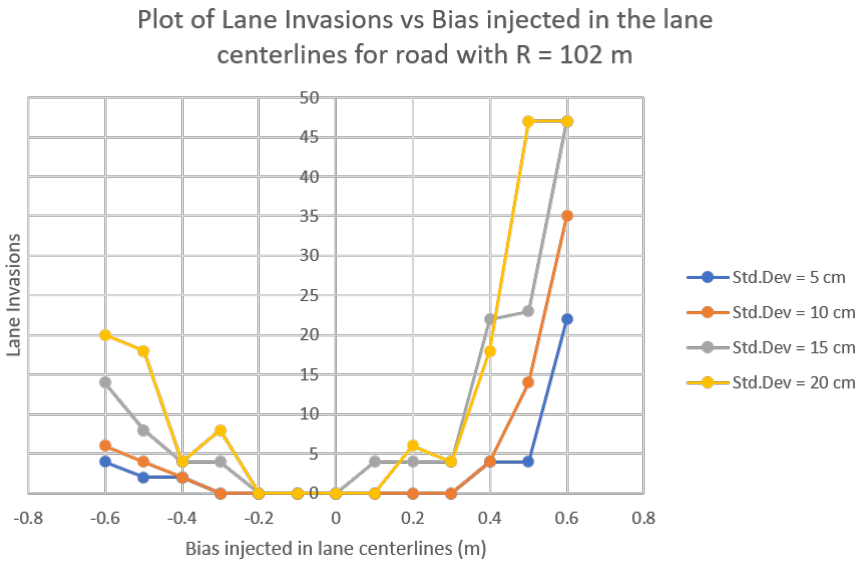


Figure 5.13: Plot of Lane invasions vs Bias injected in lane centerlines for road with 102 meters curvature (m).

In Figure 5.12, we observe the rise of mean absolute error with increasing bias present in the noise signal injected in the route. The mean absolute error peaked at a bias and jitter of -0.60 and 0.20 meters, respectively. The behaviour of mean absolute error with respect

to bias remained the same despite an increase in jitter. In Figure 5.13, the number of lane invasions increase at higher orders of bias in the noise signal injected in the route. At higher values of jitter, the number of lane invasions increased at lower values of bias. Maximum lane invasions were observed at a bias and jitter of 0.60 and 0.20 meters, respectively. Figures 5.7, 5.10, and 5.13 illustrate a vertical parabolic behaviour of lane invasions with respect to bias injected in the lane centerlines. Lane invasions increase when the bias injected in the map is raised beyond -0.30 or 0.30 meters.

In this section, we have observed the linear relationship of mean absolute error with the bias injected in the lane centerlines. Lane invasions remain constant within a defined range of -0.2 and 0.2 meters, after which it increases with increasing bias. The increase in jitter present in the noise signal led to an increase in lane invasions. However, the increase in jitter does not have a similar impact on the mean absolute error.

## 5.4 Variation of vehicle models and bias injected in lane centerlines

In this section, the tests conducted in the previous sections are performed on 3 different models of vehicles. This was done to benchmark the performance of the vehicle selected in the previous sections and to observe the behaviour of different types of vehicles. The three categories of vehicles considered for this application were compact, mid-size, and multi-purpose [52]. The list of available vehicles in CARLA was filtered and the three vehicles mentioned below were shortlisted for conducting the required tests. Results are presented for a given vehicle model selected from CARLA blueprint library [43]. Simulations have been conducted for three different vehicle models, which are as follows:

1. Toyota Prius
2. Tesla Cybertruck
3. Tesla Model 3

Each vehicle was tested in 2 scenarios. The vehicle was driven on a straight road and on a road with a defined radius of curvature. The mean absolute error and lane invasions for each case has been presented with respect to the bias injected in the vehicle's route. The jitter was set to a fixed value to draw a comparison between noise injection performed in different scenarios with the same noise signal.

1. **Straight road**

   Gaussian noise with a varying bias and jitter of 0.15 meters was injected in the lane centerlines. The speed of the vehicle was set to 30 kph. The route taken by the vehicles is illustrated in Figure 5.2.

   In Figure 5.14, we observe the mean absolute error increased with increasing bias injected in the lane centerlines. The same behaviour was observed in all vehicle models. Cybertruck had the maximum mean absolute error at a bias of 0.60 meters. However, the peaks of mean absolute error for all three vehicle is comparable.

   In Figure 5.15, lane invasions increase with increasing bias injected in the lane centerlines. The number of lane invasions remain constant till a bias of -0.3 or 0.3 meters was exceeded. The Cybertruck encountered the highest number of lane invasions at a bias of -0.4 meters. On a straight road, the Cybertruck had the highest mean absolute error and the most lane invasions amongst the three vehicles.

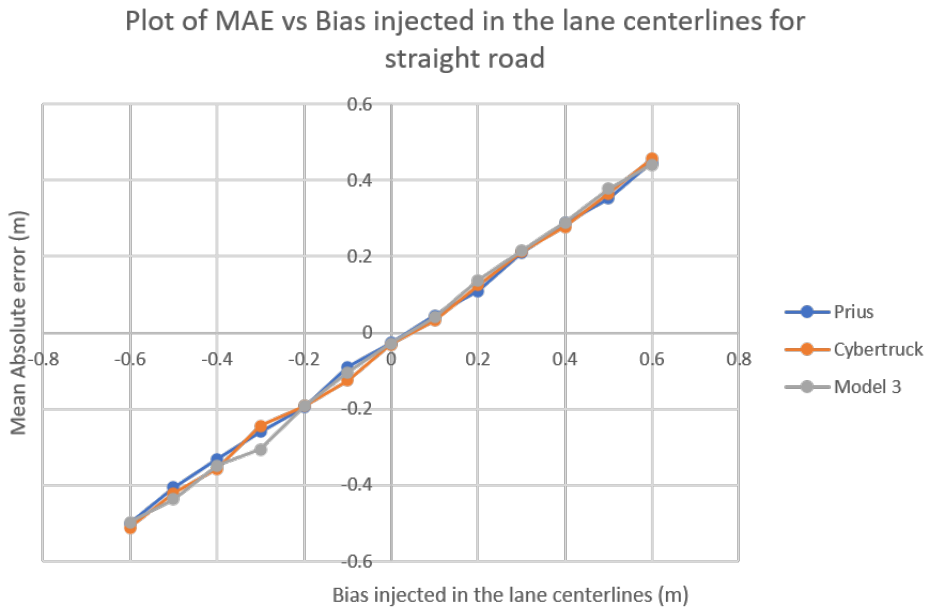


Figure 5.14: Plot of Mean Absolute error (MAE) vs Bias injected in lane centerlines (m) for straight road for 3 different vehicles.

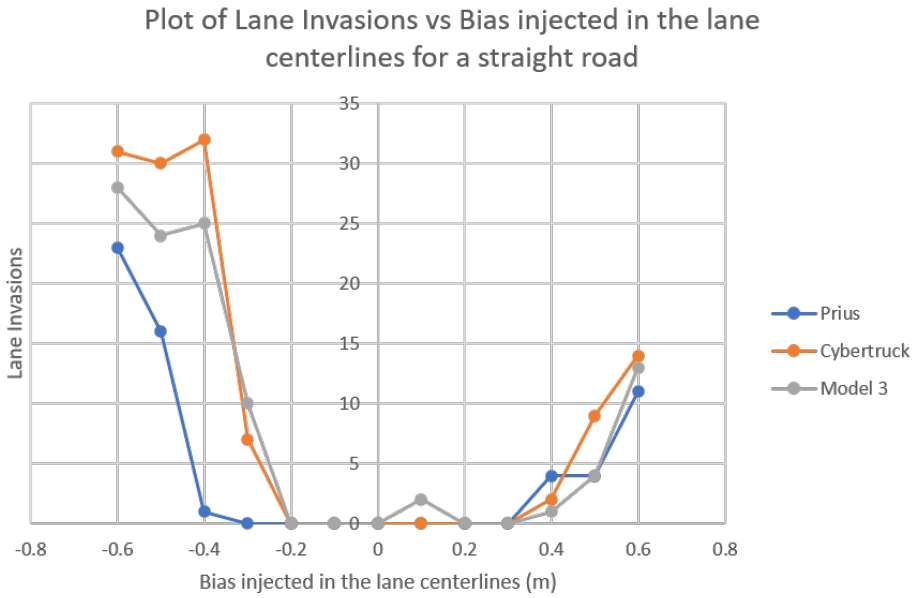


Figure 5.15: Plot of Lane Invasions vs Bias injected in lane centerlines for straight road for 3 different vehicles.

2. **Roads with defined radius of curvature**

   In the second scenario, the selected vehicle models were driven on routes with a defined radius of curvature. The routes were illustrated in Section 5.3. The results obtained from the simulation conducted are presented below.

(a) **Radius of curvature = 62 m**

Gaussian noise with a varying bias and a jitter of 0.15 meters was injected in the route followed by the vehicles. The route taken by the vehicles is illustrated in Figure 5.8. The speed of the vehicle was set to 30 kph.

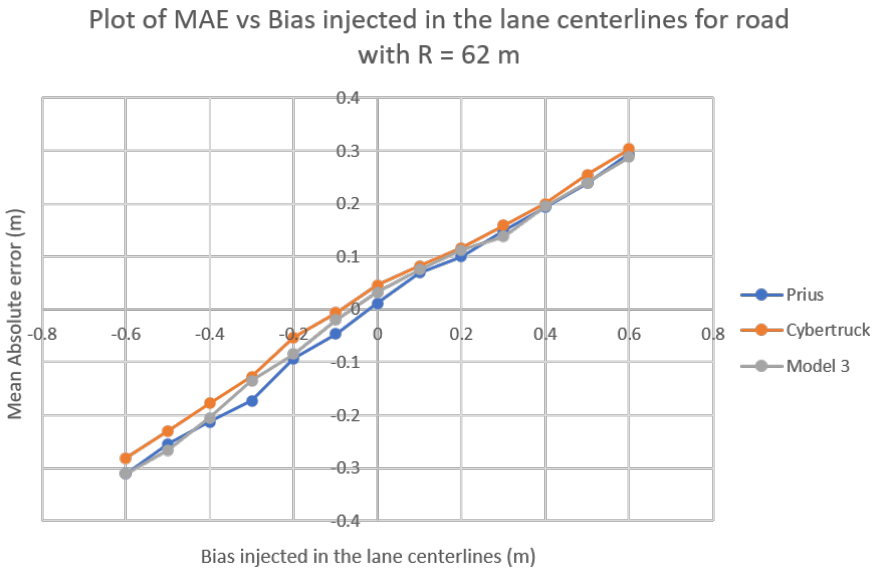


Figure 5.16: Plot of Mean Absolute error (MAE) vs Bias injected in lane centerlines for road with 62 meters curvature (m) for 3 different vehicles.

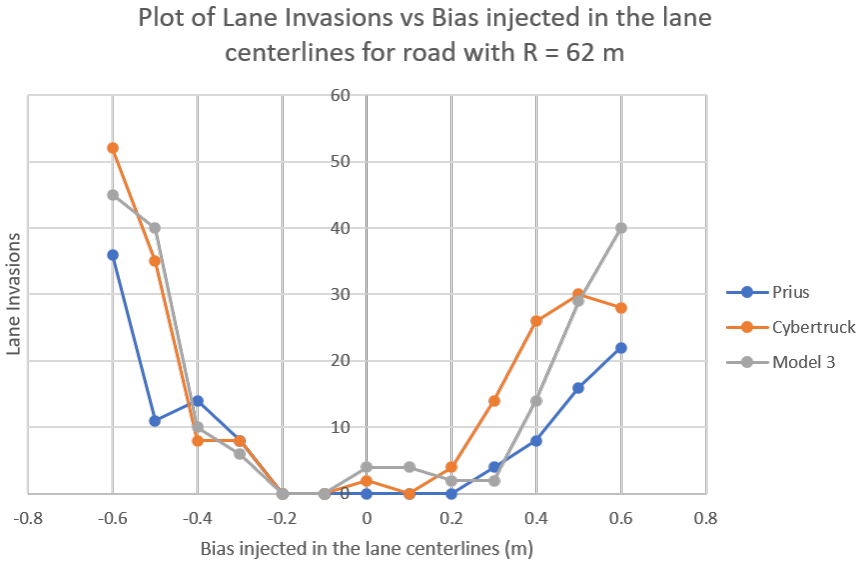


Figure 5.17: Plot of Lane Invasions vs Bias injected in lane centerlines for road with 62 meters curvature (m) for 3 different vehicles.

In Figure 5.16, mean absolute error increases with increasing bias. In the left half (negative bias) of the plot, an offset was observed in the mean absolute error, with the highest being Model 3 followed by Cybertruck and Prius. In the right half (positive

bias) of the plot, we observed an overlap in the mean absolute error for all three vehicles. Mean absolute error for Cybertruck peaked at a bias of -0.6 meters.

In Figure 5.17, lane invasions increase with increasing bias. Lane invasions remained constant in the range between -0.2 and 0.2 meters of bias. Cybertruck and Model 3 yielded the highest number of lane invasions at a bias of -0.6 and 0.6 meters respectively.

(b) **Radius of curvature = 102 m**

Gaussian noise with a varying bias and a jitter of 0.15 meters was injected in the route followed by the vehicles. The route taken by the vehicles is illustrated in Figure 5.11. The speed of the vehicle was set to 30 kph.

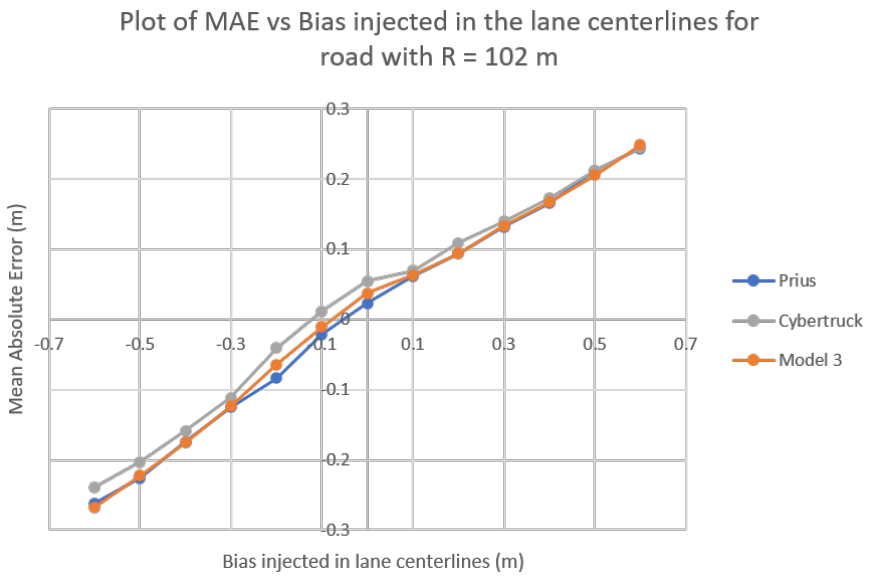


Figure 5.18: Plot of Mean Absolute error (MAE) vs Bias injected in lane centerlines for road with 102 meters curvature (m) for 3 different vehicles.

In Figure 5.18, mean absolute error increases with an increase in bias. This behaviour was observed for all three vehicle models taken into consideration. In the left half of the plot, the mean absolute errors for each vehicle are at an offset, with Model 3 producing the highest values. In the positive half of the plot, the curves for each of the vehicle models merge. However, at the peaks of bias injected in the lane centerlines, Model 3 yielded the highest mean absolute error.

In Figure 5.19, higher lane invasions are observed for a positive bias injected in the lane centerlines. The number of lane invasions remained relatively constant within the range of -0.2 to 0.2 meters. Maximum lane invasions were observed at a bias of 0.6 meters for Model 3.

In this section, we have observed that the behaviour of vehicles with respect to the increase in bias remains the same. Cybertruck yielded the highest mean absolute error on a straight road, but in the second scenario, Model 3 produced a higher mean absolute error. The number of lane invasions in the second scenario exceeded those recorded in the first scenario. Furthermore, we observed the formation of the U-pattern in lane invasions when the bias in the lane centerlines was increased.

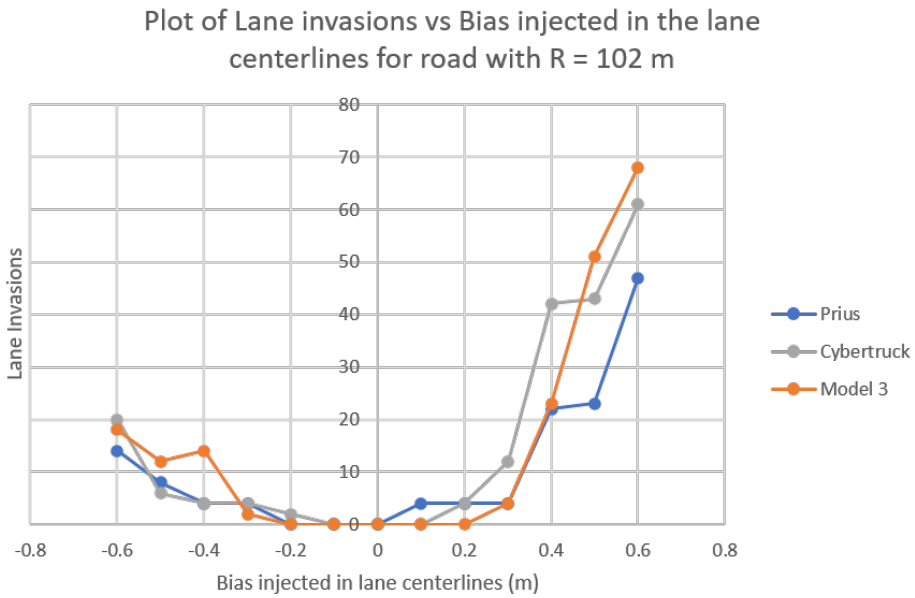


Figure 5.19: Plot of Lane Invasions vs Bias injected in lane centerlines for road with 102 meters curvature (m) for 3 different vehicles.

## 5.5 Variation of sensitivity with respect to bias injected in lane centerlines

The preceding sections presented the impact of bias on the mean absolute error and lane invasions. Vehicle models were varied to observe the impact of bias on their performance. Sensitivity was estimated for a range of bias and a fixed value of jitter. The vehicle of selection, Prius, was driven on routes with different radius of curvatures. In this section, the sensitivity of mean absolute error with respect to different curvatures has been presented. Initially, lateral noise was injected in the lane centerlines. Furthermore, combinations of noise injected in the lateral and longitudinal direction were investigated. Each combination was tested using a fixed jitter component while varying the radius of curvature. Combinations of noise injection which have been applied to the lane centerlines are as follows:

1. *Lateral noise injection*
2. *Lateral noise and longitudinal bias*
3. *Noise in lateral and longitudinal direction*

Figure 5.20 presents the estimated sensitivity when noise is injected only in the lateral direction of the route. At a small radius of curvature (6.5 m), the vehicle at 0.2 meters jitter has the highest sensitivity. At higher radii of curvature, sensitivity remained the same at different values of jitter. The sensitivity observed at 58 and 62 meters is relatively higher than that observed at 102 meters.

Figure 5.21 presents the estimated sensitivity for different combinations of noise injection in the lane centerlines. The jitter was set to 0.1 meters. Maximum sensitivity was observed when noise is injected in both the lateral and the longitudinal directions. The case with lateral noise and bias in the longitudinal direction yielded a sensitivity that is relatively close to the former. The output for the two cases almost overlap each other at higher radii of curvature. When lateral noise was injected in lane centerlines, the least sensitivity was obtained.

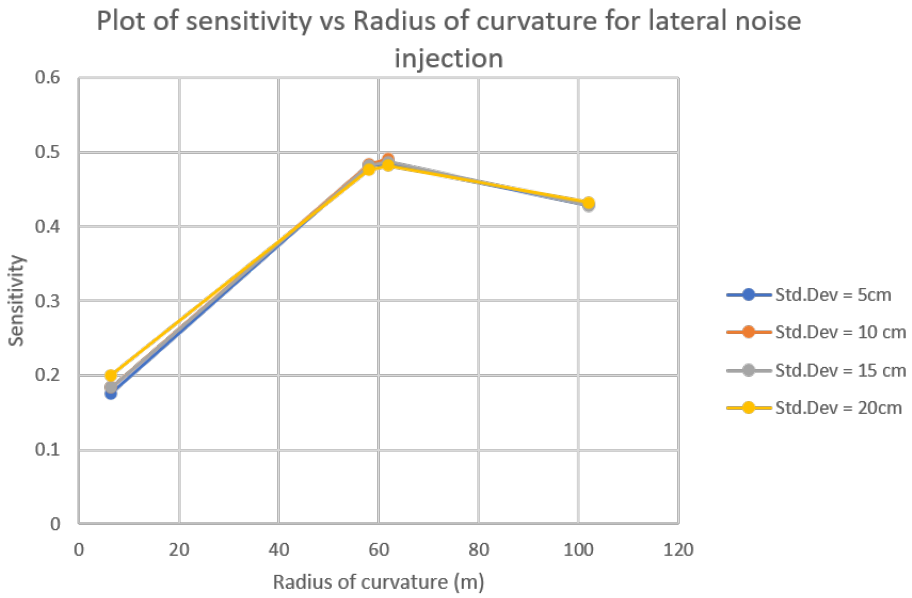


Figure 5.20: Plot of Sensitivity vs Radius of curvature (m) for jitter.

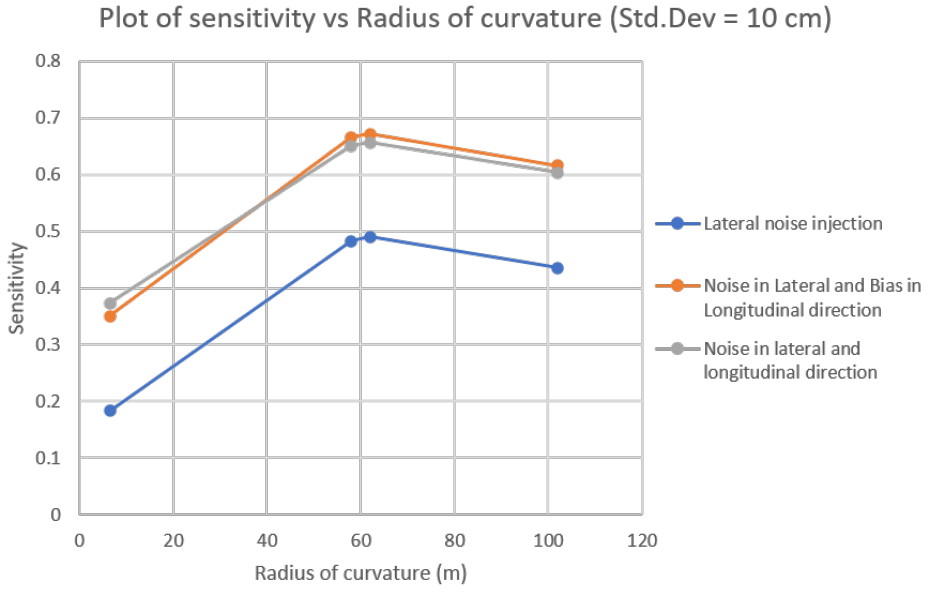


Figure 5.21: Plot of Sensitivity vs Radius of curvature (m) for different combinations of noise injection with a jitter of 0.1 meters.

Figure 5.22 presents the estimated sensitivity for different combinations of noise injection in the lane centerlines at a jitter of 0.2 meters. Maximum sensitivity was observed when a lateral noise and a longitudinal bias were injected in the map. Noise in both lateral and longitudinal directions yielded a marginally lower sensitivity as compared to the previous case. At a radius of 6.5 meters, the sensitivity remains the same for both cases however the two cases drift apart at higher radii of curvature. Lateral noise injection yielded the lowest sensitivity amongst the three

considered combinations.

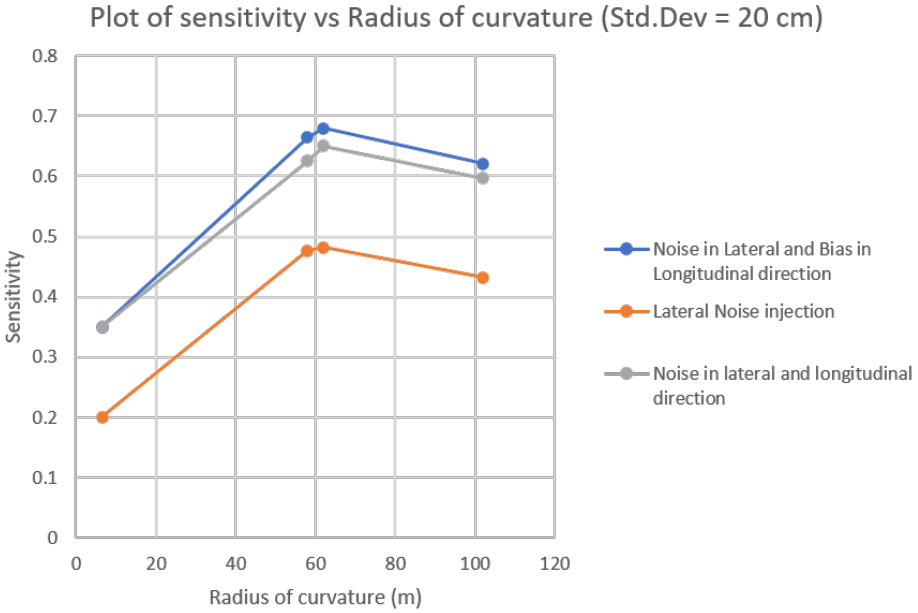


Figure 5.22: Plot of Sensitivity vs Radius of curvature (m) for different combinations of noise injection with a jitter of 0.20 meters.

In this section, we have observed the effect of injecting noise in the vehicle's route on the sensitivity of the mean absolute error. The case consisting of lateral and longitudinal noise injection yielded the highest sensitivity when the jitter was set to 0.1 meters. However, the case where lateral noise coupled with longitudinal bias yielded the highest sensitivity when the jitter was set to 0.20 meters. The case with only lateral noise injection yielded the least sensitivity amongst the three considered cases at both the values of jitter.

# Chapter 6

# Discussions

In this chapter, the results which have been presented in the previous two chapters are discussed. Key results are identified and analysed further to garner a conclusion.

## 6.1 Safety analysis findings

The safety analysis was performed from the point of view of map manufacturers, to develop an understanding regarding the usage of maps by autonomous vehicles. The architecture of the autonomous driving system was selected considering the current state of autonomous driving in the automotive market. Stakeholders were identified, and their respective losses were listed out, with an impetus being on the losses concerning the quality of map provided and the vehicle performance. Four primary stakeholders were determined from the problem statement, and 11 losses were ascertained. Hazards were identified in this process which may occur in the standard operation of the system. These hazards could have gone unnoticed in conventional safety analysis since they do not occur due to the failure of any given subsystem. Eight potential hazards were identified and their respective system constraints were set.

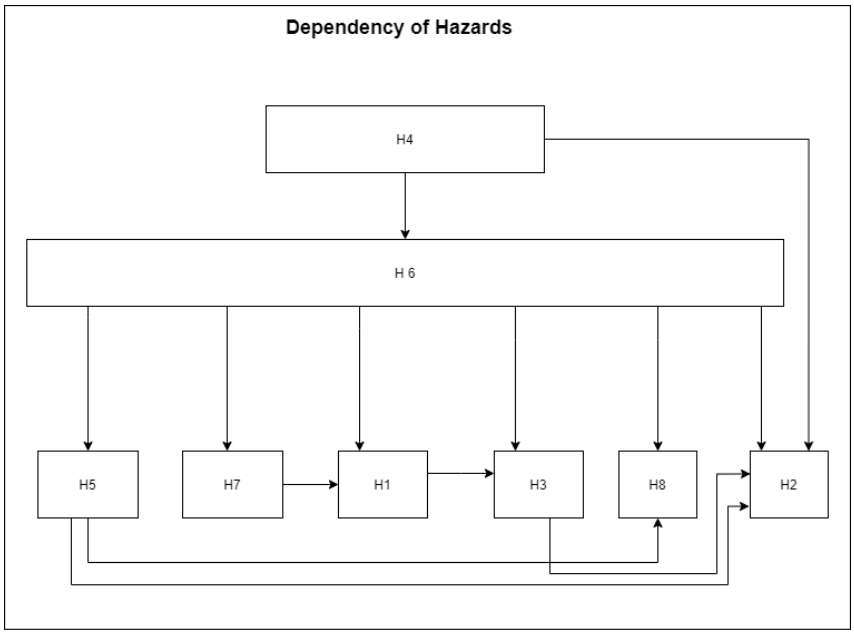


Figure 6.1: Hazard dependency chart

Figure 6.1 illustrates the dependency of hazards identified from the safety analysis performed on the automated driving vehicle. The hazards are presented in Table 4.2. A dependency was established between the hazard H3 and hazards H2, H7, and H1. H3 occurs when the vehicle follows an incorrect trajectory. If the vehicle follows the wrong trajectory, the vehicle could approach an area that exceeds the Operational design domain (ODD) of the AD system. The AD mode may remain activated in a restricted area, which results in hazard H2. The vehicle following an incorrect path could result in the erratic performance of the vehicle. Thus, hazard H3 could cause the H7. Moreover, H3 results in the violation of spacing requirements concerning neighbouring vehicles, which in turn causes hazard H1. Thus, multiple hazards may occur due to the occurrence of a single hazard, which was identified from the analysis.

Functions performed by different sub-systems were listed out to understand the role each sub-system plays in meeting the goal of automation. In addition to defining the functions, the sub-system responsible for performing the function and the subsystem requiring the output was also listed. This ensures the chain of communication between sub-systems has been reviewed sufficiently. A high-level control structure of the automated driving system was modelled, which was followed up by low-level modelling of all the required sub-systems. Eight control structures were modelled to represent the system and its subsequent sub-systems. Sixty five control actions were listed out using the modelled control structures. The control actions were filtered based on their dependence on the autonomous vehicle or the map.

The list of identified control actions were then translated to unsafe control actions using the fur different categories mentioned in Section 2.1.3. This process yielded 322 unsafe control actions. Unsafe control actions were inverted to set controller constraints on the AD system, which resulted in 383 system constraints. Unsafe control actions were filtered based on their dependency on map features, which were used in the identification of loss scenarios.

By performing the safety analysis on the autonomous vehicle, we were able to identify different aspects of the system which could cause an unsafe scenario. This process yielded 225 loss scenarios. Using the process of categorisation based on severity and exposure, the pool of scenarios was shortened to form a concise and high-priority list of 17 scenarios.

The root-cause analysis categorised 60 % of high-priority scenarios as unavoidable scenarios. The safety of the passengers in such cases can be ensured by placing a higher confidence level on the map data as compared to sensor data. To complement the higher reliance on map data, the map provided must be regularly updated and must meet the required quality. If inaccurate map data is provided to the vehicle, it could compound the existing scenario and put the vehicle into a hazardous situation. The given situation can be explained further by using a scenario identified, LS 1, which is presented in Table B.3. The vehicle encounters a foggy or low-visibility environment within which the sensors of the vehicle cannot picture the lanes at the required confidence levels. Map data consisting of lane features must be provided to the vehicle to ensure the vehicle stays on the same course. If the vehicle receives inaccurate lane features, the vehicle may shift lanes and could cause a hindrance to the vehicles in the neighbouring lanes, thus becoming a hazard.

In the above-mentioned scenario, the dependency of an autonomous driving system on the map was demonstrated. In a worst-case scenario, the dependence on map data aids in ensuring the safety of the passengers. However, complete dependence on sensors or maps for autonomous driving would not be optimal in such situations. In the loss scenario, LS 6, the vehicle received inaccurate speed limit information. This forced the vehicle to drive at a speed higher than the defined limit, which lead to the occurrence of a hazard. Thus, the vehicle must have a safe level of dependency on map features. Using the simulation platform, we have demonstrated the dependency of a vehicle's lateral control on the quality of map. This has been discussed in the next section.

## 6.2 CARLA simulation findings

Simulations were conducted on an autonomous driving vehicle in CARLA [25]. Key performance indicators (KPI) were defined and used in the evaluation of the impact of different parameters on

the vehicle's performance. In addition to the sampling size, the bias and jitter were also varied. The noise signal generated was injected in the waypoints, which are used for navigation. KPI's were observed and measured for each of the scenarios and have been presented in the previous section. The impact of uncertainty in the map was observed in two scenarios. The two key scenarios which have been tested are on a straight path and a path with a defined radius of curvature. The key results presented in the previous section regarding each KPI will be discussed.

From the study conducted on the variation of the sampling size of the map, we can ascertain that the mean absolute error (MAE) is linearly proportional to the sampling size of the map. The MAE of the vehicle decreases when the sampling size of the map is decreased. However, the change in performance of the vehicle is minimal at lower magnitudes of sampling size. The mean absolute error begins to stabilise when the sample size of the map is set to 0.25 meters. The slope of mean absolute error with respect to the sampling size remains consistent between 1.5 and 0.25 meters. On decreasing the sampling size below 0.25 meters, the slope decreases.

From the scenarios which have been tested, we observed that the mean absolute error (MAE) of the vehicle was linearly proportional to the amount of bias injected in the lane centerlines. Jitter does not exhibit a similar impact on the MAE as compared to bias. The linear relationship exhibited between MAE and bias is due to the vehicle tracking a noise injected path. Since the noise injected path was at a varying but almost constant distance from the original centerline, the vehicle's location shifts by that given distance to the original centerline. The MAE recorded was not equal to the amount of bias injected in the centerlines, since a gaussian noise signal with a bias and jitter was injected in it. The noise signal was injected into the route element-wise, thus affecting all the elements of the route.

Lane invasions exhibited a non-linear relationship with bias. There are minimal lane invasions observed within a range of -0.2 and 0.2 meters. This behaviour was observed for all three vehicle models tested. When the bias injected was increased beyond this range, lane invasions increased. Jitter injected in the lane centerlines led to a higher number of lane invasions. A higher magnitude of jitter in the noise signal led to higher lane invasions. A noise signal with a higher magnitude of jitter resulted in the vehicle having snappy steering. The vehicle drove in a wavy pattern, which resulted in more lane invasions being recorded. If lower magnitudes of jitter were coupled with higher magnitudes of bias, a high number of lane invasions would be detected since the vehicle is positioned very close to the lane borders. The probability of the vehicle cutting the lane borders would be high, thus resulting in higher number of lane invasions.

Sensitivity exhibits a linear relationship with the radius of curvature. In the figures presented, we observed an increase in sensitivity with the increase in the radius of curvature. There was a slight dip in the sensitivity when the radius of curvature is set to 58 and 62 meters. However, the trend of sensitivty with respect to curvature was linear. Maximum sensitivity was observed when bias and jitter were injected on a straight road. When a vehicle drives on a straight road, there is minimal steering input to keep the vehicle heading straight. However, when noise containing a high order of bias and jitter, was injected on a straight road, the vehicle generated a high amount of steering, which resulted in a high MAE. On a path with curvature, the steering output generated by the vehicle remains relatively the same. Minor changes in steering output correspond to a lower mean absolute error, which in turn led to low sensitivity.

Thus, we have observed the impact of map uncertainty on the performance of a vehicle. Using the defined KPIs, we have evaluated the performance of different vehicles in varying scenarios. Mean absolute error was used to evaluate the lateral positioning of the vehicle while tracking a route with different noise signals. Lane invasions were used to understand the number of occasions the vehicle does not stay in its lane. A vehicle that yielded multiple lane invasions would serve as a hazard to all the vehicles in the neighbouring lanes. Thus, lane invasions have been used as a measure to evaluate the safety of noise injected in the map.

## 6.3 Limitations

The research conducted does have limitations, which have been addressed in this section. The architecture of the AD system has been selected using literature, which is simplistic. AD systems employed by OEMs may have a greater level of complexity, which could yield a larger pool of results. Each of the shortlisted simulation environments cater to the testing and development of autonomous driving systems. They are not built from the perspective of testing a state-of-the-art AD system for a given map, which serves as a limitation to this research. An autonomous driving simulator must be adapted to perform the simulations from a mapmakers' point of view, which may not necessarily be the best approach.

The selected simulation environment, CARLA, has limitations in terms of available routes present in the inbuilt maps. In this study, the maximum available radius of curvature provided in CARLA has been utilized. However, in reality, the radius of curvatures of highways are much larger in magnitude, in the order of kilometres, as compared to those provided in the simulation environment which is in the order of meters. CARLA does provide an OpenDrive standalone mode to import OpenDrive files. However, the ability to tune parameters of the road surface in the map such as friction are unavailable, which causes unstable vehicle behaviour at higher speeds. There is a lack of complementary documentation concerning these features, which makes the task tedious and challenging. CARLA provides limited functionalities regarding navigation in a defined map. CARLA provides navigation for vehicles in the form of waypoints. It does not provide any other map features, such as traffic signs, for navigation in the defined world. Thus, having only waypoints as the single source for navigating a given vehicle serves as a limiting factor.

# Chapter 7

# Conclusions and future works

In this thesis, the impact of the map on the functional safety of an automated driving vehicle was observed by performing two key tasks, the first being the safety analysis and the second being simulation of map uncertainty in a self-driving environment. The safety analysis was primarily aimed at identifying scenarios in which an AD system could encounter a hazard and thus put the safety of the vehicle at risk. Simulations were performed to visualise the impact of the uncertainty on the lateral control of the vehicle. Results of both the tasks are presented and discussed in the previous sections. During the course of the thesis, two research questions were identified, whose answers will be addressed below.

***In what scenarios does the traffic and lane features of the map impact the functional safety of automated vehicles?***

The functional safety of an automated driving (AD) vehicle is impacted by maps in a diverse set of scenarios. Loss scenarios were identified for the AD system which occurs due to unsafe interactions between different components of the system. When a given scenario occurs, the sensing unit continues to perform as required, however, the input needed to make decisions is not received. This forces the vehicle to rely more on the map to make real-time decisions. Thus, we have established a level of dependency between the vehicle and the map in a worst-case scenario.

The loss scenarios identified from the analysis occur despite the optimal functioning of each sensor. This justifies the selection of the safety analysis technique since the AD system enters a hazardous state without the failure of a given system or sub-system. The safety analysis conducted on the AD system yielded loss scenarios concerning different factors. Majority of the high-priority scenarios concerned lane features and traffic signs present in the map or on the road. Lane features and traffic sign-related loss scenarios were encountered when issues in the road infrastructure, weather, or vehicle positioning were faced. These scenarios relied on utilising the provided map data to ensure the safe movement of the vehicle. However, loss scenarios were also identified in cases wherein incorrect map data was provided to the vehicle's AD system. These scenarios occurred despite receiving optimal sensor feedback, however, the dependence on map data resulted in a hazard. Thus, complete reliance on map data when the vehicle is in its operating domain would not be optimal and safe.

Therefore, we can conclude that an automated driving vehicle cannot completely rely on either sensor data or map data. Complete reliance on either of the two could result in the occurrence of hazards, which compromises the safety of the AD system. The AD system must use a fusion of sensor and map data. In a given worst-case environment, the vehicle can rely on minimal sensor data and map features to navigate the vehicle to a safe location. However, this is dependent on the quality and accuracy of map data provided to the vehicle. The level of dependence on given map features must be established based on the availability and quality of data delivered to the vehicle. If the level of quality of map data provided is not as per requirements, a take-over request must be issued to the driver. This would ensure control of the system is handed back to the driver, thus maintaining the level of safety of the system. The Operational Design Domain (ODD) of the AD system could either be limited or broadened based on the accuracy of map data provided to

it, thereby preventing the occurrence of loss scenarios due to inaccurate data.

***In the event of camera failure, what is the dependency of lateral control of an automated driving vehicle on the quality of accurate maps?***

The aim of conducting simulations in a self-driving simulator was to observe the impact of the uncertainty of the map on the lateral performance of the vehicle. Lateral performance was measured using three key performance indicators, which were the mean absolute error (MAE), lane invasions, and sensitivity. A gaussian noise signal composed of a bias and jitter component was injected into the map to simulate uncertainty. From the initial tests conducted, we observed that the AD vehicle was able to negotiate the selected paths successfully with minimal lateral errors. Thus, we can conclude that in a given worst-case environment, a vehicle can manoeuvre a given path using minimal sensor and map data. However, this is possible only if the map is of high quality and does not have large noise signals ingested in it.

A map can never be a perfect replica of reality since reality will undergo constant change. There will always be an element of noise in the map features provided to the vehicle. However, the magnitude of noise present in the map may differ based on the quality checks conducted by mapmakers during the production and delivery of maps. Through simulations, we have observed the impact that uncertainty in a map can have on the lateral control of the vehicle. Bias, the mean of the noise, has the largest impact on lateral performance when compared to jitter. Bias has the largest impact since it forces the vehicle to shift positions laterally, which results in lower lateral performance. On the other hand, the impact of jitter is magnitude-dependent. At lower orders of jitter, there is minimal impact on the lateral performance. However, excessive jitter results in drunk-driver behaviour, which leads to lower comfort and safety for the passengers and the neighbouring vehicles.

Besides the element of noise, the sampling size of the map plays a crucial role in the lateral performance of the vehicle. The mean absolute error of the vehicle decreases linearly with the decrease in sampling size. However, the improvement in mean absolute error decreases as the sampling size is decreased beyond a given threshold. Thus, a map with a lower sampling size may yield similar mean absolute errors as compared to that obtained at a comparatively higher sampling size. Furthermore from a mapmakers' point of view, the costs that are incurred in producing a map of such sample sizes would outweigh the benefits that are obtained in lateral performance. Thus, a break-even must be established between the required lateral performance, cost of production, and the production capability of the map production system.

## 7.1 Recommendations and Future works

This project mainly focused on analysing the impact of maps on the safety of a vehicle. The scope of research conducted can be broadened to analyse the impact on not just the safety, but the overall performance of the vehicle. KPIs were defined to observe the impact of map uncertainty on the overall safety of the vehicle. KPIs can be defined which could be used to evaluate the different aspects of vehicle performance such as vehicle slip. Moreover, this could aid in enhancing the given requirements placed on the map production system.

The vehicle modelled in CARLA has minimal sensor functionalities. The given vehicle model is equipped with a functional GPS and an IMU. However, the camera and LiDAR have been deactivated to represent a worst-case scenario for the AD system. The scope of the vehicle's AD system must be broadened to include a lane assist and a cruise control system. This would facilitate the appropriate representation of a vehicle equipped with advanced driver assist systems (ADAS). The lateral control method has been selected based on given time constraints and the required performance at lower speeds. The results obtained from this study can be benchmarked by employing different lateral control strategies. Simulations have been conducted considering an inner-city application of maps, which is limited to 30 kph. The speed of the vehicle must be increased to 140 kph, which would facilitate simulating scenarios encountered on highways. This would lead to a wider range of scenarios that can be tested in the given environment.

Simulations were conducted to analyse the impact of map uncertainty on the safety of the

vehicle. A vehicle would be spawned and driven in the world, on a pre-determined route using a given lateral control algorithm. Bias and jitter were injected in the route and KPIs were recorded. The impact on traffic safety can be analysed even further by extending the application to include multiple vehicles, which are being driven in the vicinity of the ego vehicle. This would provide an opportunity to observe, first-hand, the impact of bias and jitter in the ego vehicle's path on the neighbouring vehicles. The maximum level of bias and jitter needed to induce a crash with a vehicle in the neighbouring lane can be estimated by executing an optimisation script that would utilize the parameter, lane invasions, as an input.

# Bibliography


[1] TomTom NV, "TomTom Company WebPage." 1, 19

[2] TomTom NV, "AutoStream Product." 1

[3] K. Schuerman and M. Rosikiewicz, "TomTom map production system dataflow architecture," Tech. Rep. October, TomTom NV, 2019. 1, 2

[4] TomTom NV, "AUTOSTREAM SDK HD Map Data Model and Attributes," tech. rep., 2019. 1, 2

[5] TomTom NV, "TomTom: How we make our HD maps," 2020. 1

[6] TomTom NV, "Floating Car Data : Upload specification Table of Contents," Tech. Rep. December, TomTom NV, 2019. 1

[7] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car-part i: Distributed system architecture and development process," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 7131–7140, 2014. 3, 4

[8] T. G. Reid, S. E. Houts, R. Cammarata, G. Mills, S. Agarwal, A. Vora, and G. Pandey, "Localization requirements for autonomous vehicles," *arXiv*, no. February 2020, 2019. 3

[9] N. G. Leveson and J. P. Thomas, "STPA Handbook," tech. rep., MIT, 2018. 3, 7, 8, 9, 10, 11, 19, 20

[10] S. M. Sulaman, A. Beer, M. Felderer, and M. Host, "Comparison of the FMEA and STPA safety analysis methods-a case study," *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)*, vol. P-292, pp. 175–176, 2019. 3

[11] FAA, "FAA System Safety Handbook, Chapter 9: Analysis Techniques," *FAA System Safety Handbook*, 2000. 3

[12] V. H. Balgos, *A Systems Theoretic Application to Design for the Safety of Medical Diagnostic Devices*. PhD thesis, Massachusetts Institute of Technology (MIT), 2012. 3

[13] C. Becker, L. Yount, S. Rosen-Levy, J. Brewer, and NHTSA, "Functional Safety Assessment of an Automated Lane Centering System," Tech. Rep. August, NHTSA (National Highway Traffic Safety Administration), US Department of Transportation, Washington, DC, 2018. 3, 4, 8

[14] H. Elrofai, J.-P. Paardekooper, E. de Gelder, S. Kalisvaart, O. op Den Camp, and TNO, "Streetwise: Scenario-based safety validation of connected and automated driving," tech. rep., TNO, Helmond, Netherlands, 2018. 3

[15] N. Silvis-Cividjian, W. Verbakel, and M. Admiraal, "Using a systems-theoretic approach to analyze safety in radiation therapy-first steps and lessons learned," *Safety Science*, vol. 122, no. October 2019, p. 104519, 2020. 3, 8, 10

[16] EEnewsautomotive, "Number of automotive ECUs continues to rise," 2019. 3

[17] SAE International, "SAE Levels of automation." 4, 19, 27, 29

[18] G. Bagschik, A. Reschka, T. Stolte, and M. Maurer, "Identification of potential hazardous events for an Unmanned Protective Vehicle," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2016-Augus, pp. 691–697, 2016. 8

[19] K. Allenby and T. Kelly, "Deriving safety requirements using scenarios," in *Proceedings of the IEEE International Conference on Requirements Engineering*, no. May, pp. 228–235, 2001. 11

[20] International Organization for Standardization, "ISO 26262-3 - Road vehicles — Functional safety — Concept phase," *NEN-ISO 26262-3:2019*, vol. 1, 2011. 12, 19

[21] R. S. Martínez, *System Theoretic Process Analysis of Electric Power Steering for Automotive Applications*. PhD thesis, Massachusetts Institute of Technology (MIT), 2015. 12

[22] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors (Switzerland)*, vol. 19, no. 3, pp. 1–29, 2019. 12

[23] S. van Schouwenburg, *Evaluating urban dynamic SLAM in an environment*. Master thesis, Delft Univeristy of Technology (DUT), 2019. 12

[24] TomTom NV, "TomTom: Autonomous driving simulator evaluation." 13

[25] CARLA, "CARLA homepage." 13, 14, 24, 62

[26] O. S. Robotics, "Gazebo." 13

[27] Microsoft, "AirSim." 13

[28] Deepdrive, "Deepdrive Self-driving AI." 13

[29] MIT and Udacity, "Udacity Self-driving car." 13

[30] M. Martinez, C. Sitawarin, K. Finch, L. Meincke, A. Yablonski, and A. Kornhauser, "Beyond Grand Theft Auto V for Training, Testing and Enhancing Deep Learning in Self Driving Cars," pp. 1–15, 2017. 13

[31] C. H. University of North Carolina and U. o. C. Florida, "AutonoVi." 13

[32] Mechanical Simulation Corporation and MathWorks, "CarSim." 13

[33] IPG Automotive, "CarMaker." 13

[34] AiMotive, "aiSim." 13

[35] M. Software, "Virtual Test Drive (VTD)," 13

[36] TASS International, "PreScan." 13

[37] Cyberbotics, "Webots." 13

[38] LG Electronics America, "LGSVL Simulator." 13

[39] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," no. CoRL, pp. 1–16, 2017. 14

[40] CARLA, "CARLA Python API." 14, 15, 16, 21, 46

[41] Association for standardization of Automation and Measurement systems (ASAM e.v), "OpenDrive," 2021. 14, 22

[42] CARLA, "CARLA docs: Maps and Navigation." 14, 15, 16, 22

[43] CARLA, "List of available vehicles," 2021. 15, 53

[44] R. C. Conlter and CMU, "Implementation of the Pure Pursuit Path 'hcking Algorithm," Tech. Rep. January, The Robotics Institute Camegie Mellon University, Pittsburgh, Pennsylvania 15213, 1992. 17

[45] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-e. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. V. Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley : The Robot that Won the DARPA Grand Challenge," vol. 23, no. April, pp. 661–692, 2006. 17

[46] A. Patnaik, M. Patel, V. Mohta, H. Shah, S. Agrawal, A. Rathore, R. Malik, D. Chakravarty, and R. Bhattacharya, "Design and Implementation of Path Trackers for Ackermann Drive based Vehicles," 17

[47] A. L. Rankin, C. D. Crane III, and D. G. Armstrong II, "Evaluating a PID, pure pursuit, and weighted steering controller for an autonomous land vehicle," *Mobile Robots XII*, vol. 3210, pp. 1–12, 1998. 17

[48] M. Samuel, M. Hussein, and M. Binti, "A Review of some Pure-Pursuit based Path Tracking Techniques for Control of Autonomous Vehicle," *International Journal of Computer Applications*, vol. 135, no. 1, pp. 35–38, 2016. 17

[49] M. Ding Yan, "Three Methods of Vehicle Lateral Control: Pure Pursuit, Stanley and MPC," 2020. 18

[50] International Organization for Standardisation, "Road Vehicles — Terms and Definitions of Test Scenarios for Automated Driving Systems," *ISO 34501*, vol. 7, no. ISO/TC 22 / SC 33, p. 10, 2021. 20, 40, 86

[51] International Organization for Standardization, "Geographic information — Data quality — Part 1: General requirements," *ISO/CD 19157-1:2021(E)*, no. ISO TC 211/SC /WG 9, 2021. 21

[52] International Organization for Standardisation, "Road vehicles — Types — Terms and definitions," *ISO 3833:1977*, 2019. 53

# Appendix A

# CARLA

## A.1 Vehicle Coordinate system

Figure A.1 illustrates the co-ordinate system for a vehicle in CARLA. This coordinate system is used as a reference for generating a steering input.

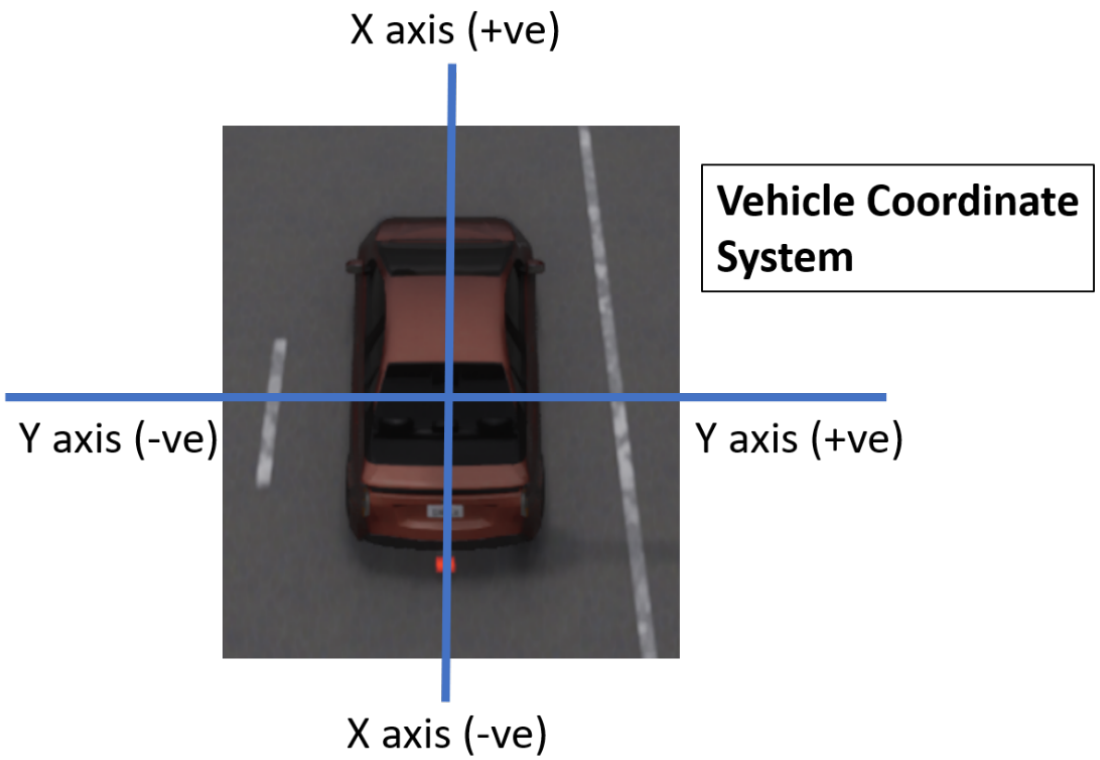


Figure A.1: Coordinate system for the vehicle

## A.2 Map coordinate system

Figure A.2 illustrates the co-ordinate system for a world in CARLA. The top view of the map has been used to present the coordinate system of the world. This coordinate system is used as a reference for injecting a noise in the waypoints, which are used for navigation by the vehicle.

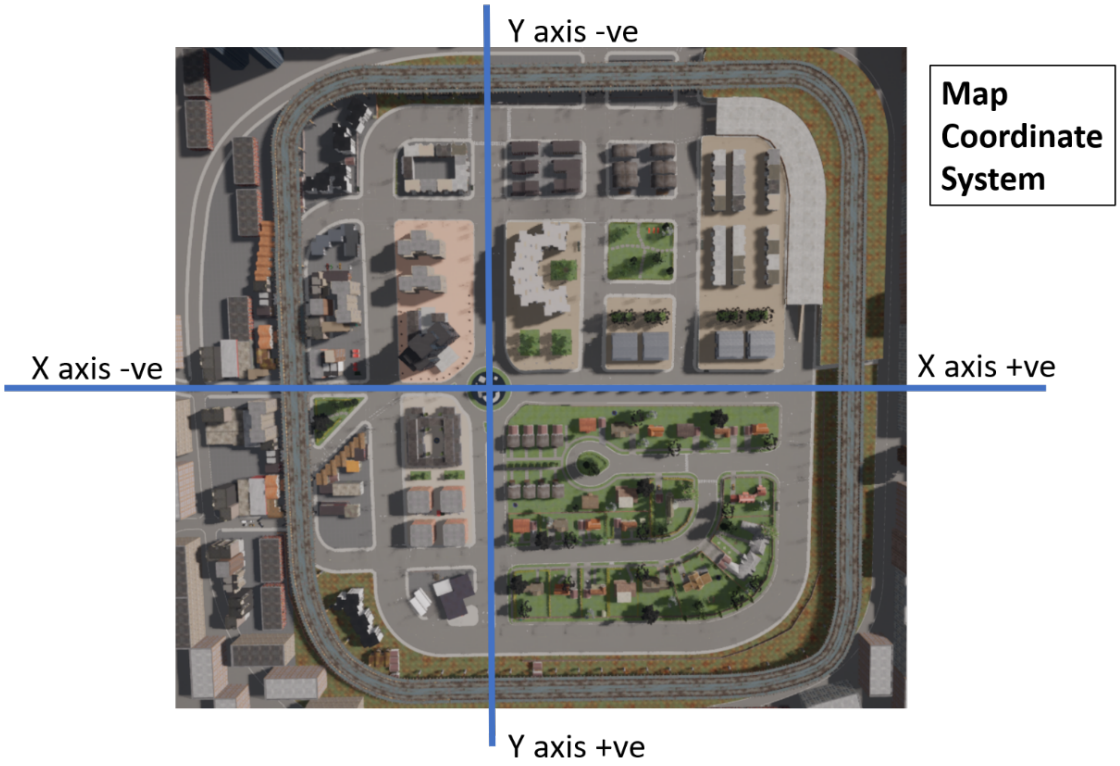


Figure A.2: Global coordinate system for the map (Top view).

## A.3 Pure Pursuit controller

In this section, the algorithm applied in the implementation of Pure Pursuit controller is presented in the form of a flow chart.

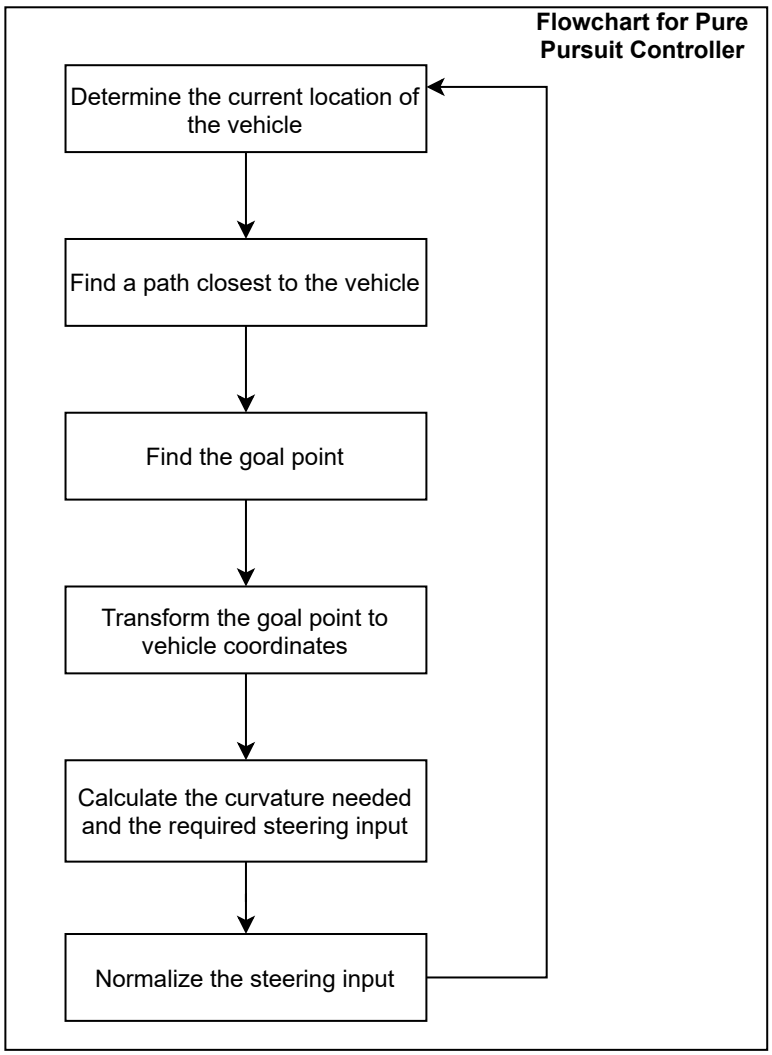


Figure A.3: Process flow chart for the pure pursuit controller.

# Appendix B

# STPA Results

## B.1 List of system constraints

Table B.1 presents the list of system constraints identified from the list of hazards. These are identified in the first step of the safety analysis conducted on the Level 2/3 Automated driving Vehicle (AV). The constraints are at a high-level and are aimed at preventing the occurrence of hazards due to unsafe interactions at the highest level of the system. In the table, the constraints defined for given hazards (in square brackets) have been listed below.

Table B.1: List of system constraints identified from STPA.

| SC ID | System constraint |
|---|---|
| 1 | The AV must maintain spacing between neighboring vehicles based on defined requirements.[H1] |
| 1.1 | A. The AV must maintain the required speed based on the speed limit data obtained from the HD maps.[H1.1] |
| 1.1 | B. The AV must follow a uniform acceleration profile to maintain the required levels of comfort for the passengers of the vehicle. [H1.1] |
| 1.2 | The AV must initiate quick deceleration in the case of an emergency. [H1.2] |
| 2.1 | The AV must engage autonomous mode only in regions/ road areas specified in the HD map.[H2] |
| 2.2 | In case of an error in mode detection, the AV must initiate a takeover request (TOR) immediately.[H2] |
| 3 | The AV must follow the path created from the fusion of the HD map and observational data.[H3] |
| 3.1 | A. The AV must communicate Floating car data to the map producer regarding wrong curvature detection. [H3.1] |
| 3.1 | B. The AV must generate a new path based on observational data to a safe state.[H3.1] |
| 3.2 | The AV must initiate emergency braking or a TOR to prevent collision with other cars/ pedestrians.[H3.2] |
| 4.1 | The AV must preload map data in its cache to protect the vehicles in case of a sudden loss of data connectivity. [H4] |
| 4.2 | The AV must have multiple data connections in place as a fail-safe measure.[H4] |
| 5.0 | A. The AV must issue TORs only if observations needing such action are drawn. [H5] |
| 5.0 | B. The AV should issue a TOR if the automated system is exiting the specified automated driving area. [H5] |

Table B.1: List of system constraints identified from STPA.

| SC ID | System constraint |
|---|---|
| 5.1 | The AV must issue a TOR only if the criterion has been satisified. [H5.1] |
| 5.2 | The mode transition of the vehicle must be performed with least amount of delay.[H5.2] |
| 6.0 | A. The AV must be parked in a safe state immediately.[H6] |
| 6.0 | B. The AV must issue a TOR to the driver if the visualization system cannot be activated again. [H6] |
| 6.1 | The AV must issue a TOR to the driver as soon as possible. [H6.1] |
| 6.2 | A. The AV must issue an error message followed by a TOR to the driver. [H6.2] |
| 6.2 | B. The AV must be parked in a safe state in case TOR is rejected by the driver. [H6.2] |
| 7 | The AV must slow down to a velocity at which the system can drive uniformly. [H7] |
| 7.1 | A. The AV must utilize the defined speed limit data provided in the HD maps. [H7.1A] |
| 7.1 | B. If speed limit data is unavailable, the AV must drive at a system defined safe speed. [H7.1] |
| 7.2 | The AV must initiate a TOR to the driver within the given timeframe if control cannot be established. [H7.2] |
| 7.3 | The AV must utilize the lane features provided within the HD maps to establish lateral control. [H7.3] |
| 7.4 | A. The AV must stop instantaneously only when an obstacle is detected. [H7.4] |
| 7.4 | B. The AV must stop instantaneously when the emergency brakes have been activated by the driver/ autonomous driving system. [H7.4] |
| 8 | The vehicle must have a defined safe mode, in which the vehicle parks itself in a safe state when such a situation is detected. [H8] |
| 8.1 | If the vehicle is stuck in the autonomous mode, it must be set to a safe state, followed by a system reboot. [H8.1] |
| 8.2 | The autonomous mode must remain disabled till the vehicle has been switched off and a complete reboot has been performed. [H8.2] |

## B.2 Control structures

Figure B.1 presents the control structure for the environment. The environment can be split into the following categories; road infrastructure, weather, terrain, and water bodies. Weather can be used to describe the state of atmosphere. This is crucial for autonomous driving systems. Foggy conditions are very difficult to navigate through due to a reduction in levels of visibility. Water bodies will also be present in an environment along which bridges, and passes have been built. Road infrastructure consists of roads, traffic and surrounding infrastructure. Roads can be categorized into highways, local roads, and intersections. The characteristics of roads such as lane groups, types, borders, trajectory and centerlines are also considered. Traffic consists of vehicles occupying lanes, pedestrians, and traffic signs and lights. Surrounding infrastructure includes overhead structures such as bridges, tunnels and other structures or landmarks such as buildings and pavements.

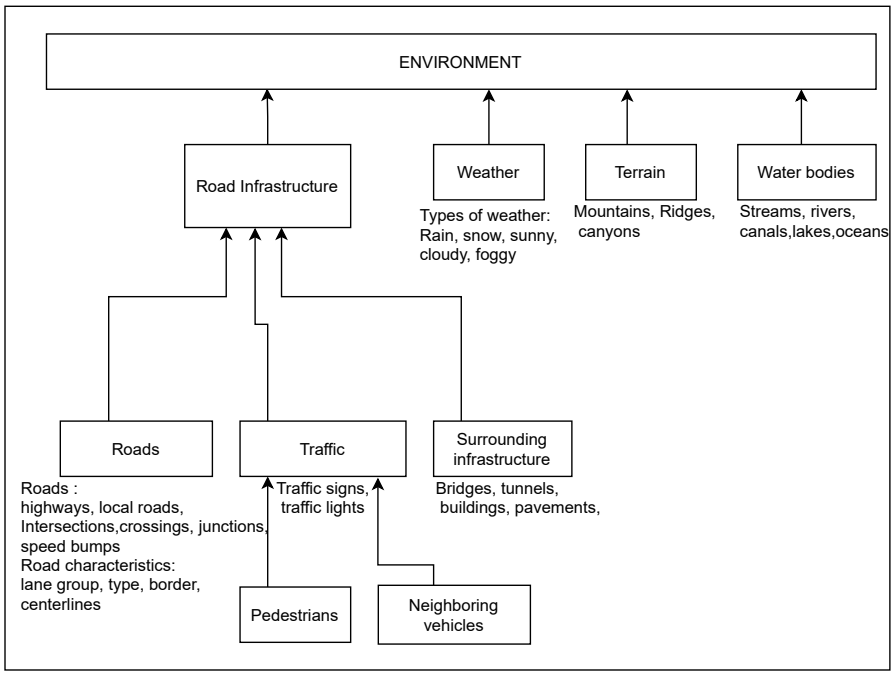


Figure B.1: Control structure of environment

Figure B.2 illustrates the perception system. Its main function is to identify physical objects such as obstacles, neighboring vehicles, overhead structures, traffic signs, and lights. The data storage unit contains the HD MAP and observational data. Data is requested from the DSU by the perception algorithm. The data is utilized for performing the following functions:

1. Detect surrounding structures using HD map data
2. Detect traffic signs and lights
3. Detect neighboring vehicles

Incoming obstacles are communicated to the control unit to activate the collision avoidance system. Furthermore, regions with detected overhead structures are utilized for restricting autonomous driving functionality.

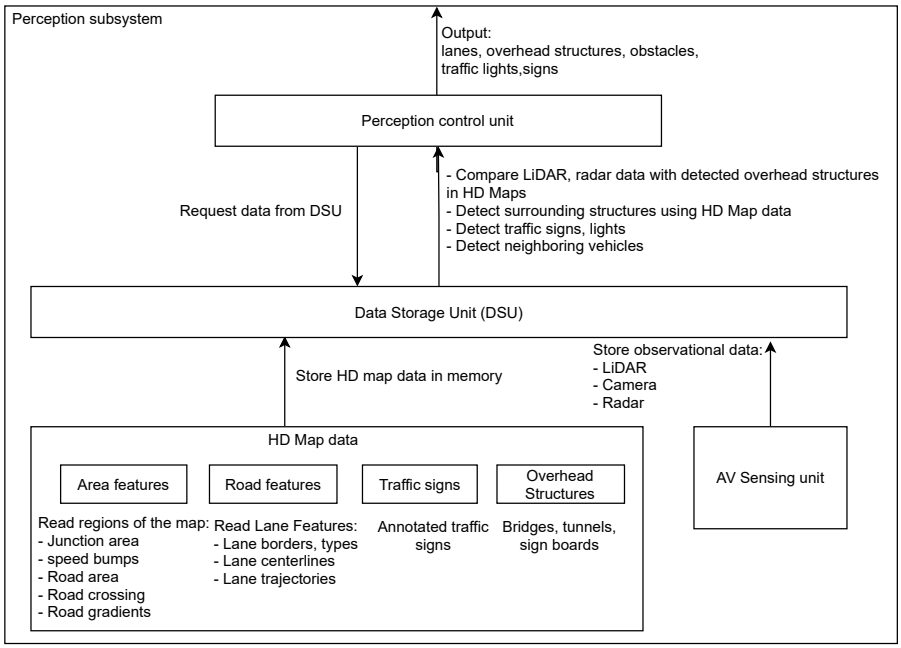


Figure B.2: Control structure of perception system

Figure B.3 illustrates the control structure employed in the Localization system. Control is established from the localization algorithm. Data from the HD map is pooled into the data storage unit. The data storage unit is shared between the localization and perception system. By pooling all the data into one memory unit, the need for multiple memory units can be eliminated. Observational data from the mounted sensors are gathered in the perception sub-unit. This data is communicated to the data storage unit in the localization system. IMU and GPS data is utilized to estimate the vehicle's current pose. It is also used to estimate the current location of the vehicle with respect to the detected lanes. The coordinates of the vehicle are communicated to the path-planning system.

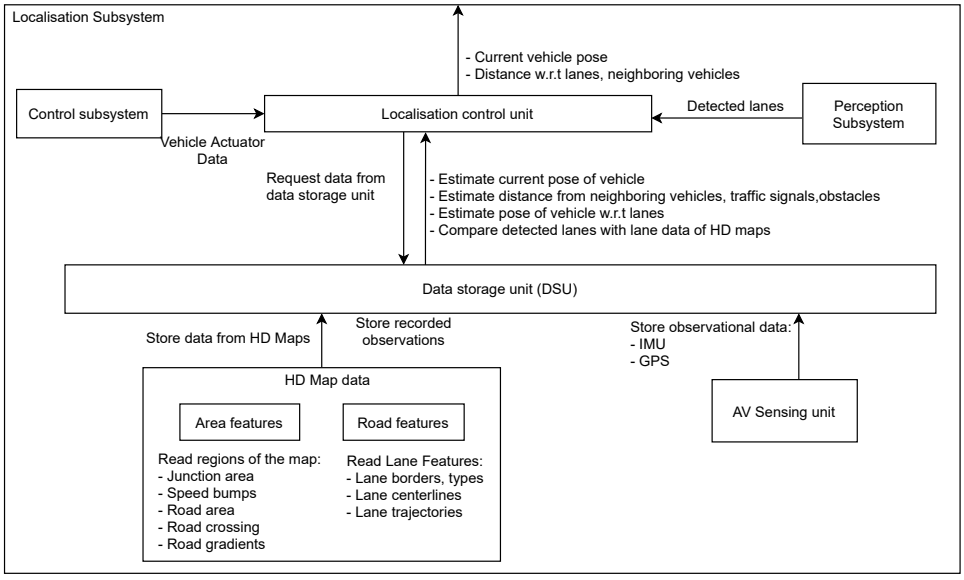


Figure B.3: Control structure of localization system

Figure B.4 presents the control structure of the vehicle control system. The vehicle control system comprises of the motion control unit and the control algorithm. The control algorithm receives mode of operation, detected obstacle data, generated path and HD map data in the form of traffic signals, lights and speed limits as inputs. It yields control commands which is fed to the motion control unit. The motion control unit is responsible for driving the vehicle. This unit receives inputs from the control unit which triggers the subsequent elements of the system. The Motion control unit (MCU) has been simplified into two forms of control; lateral and longitudinal control. Lateral and longitudinal control is utilized for controlling the lateral and longitudinal dynamics of the vehicle respectively. The actuators can be split into four subsystems; power unit, transmission, brakes, and steering. The power unit is responsible for generating the required torque output based on control inputs. The torque is fed to the transmission system to obtain the target speed. The speed may be limited based on speed limit data received from the HD map. The speed can also be restricted based on the expected road curvature. The brakes can be engaged or disengaged based on the maneuvering of the vehicle. The detected obstacle data will be used for triggering the emergency braking system (EBS). Furthermore, steering can be engaged based on the vehicle path pre-determined in the maps.

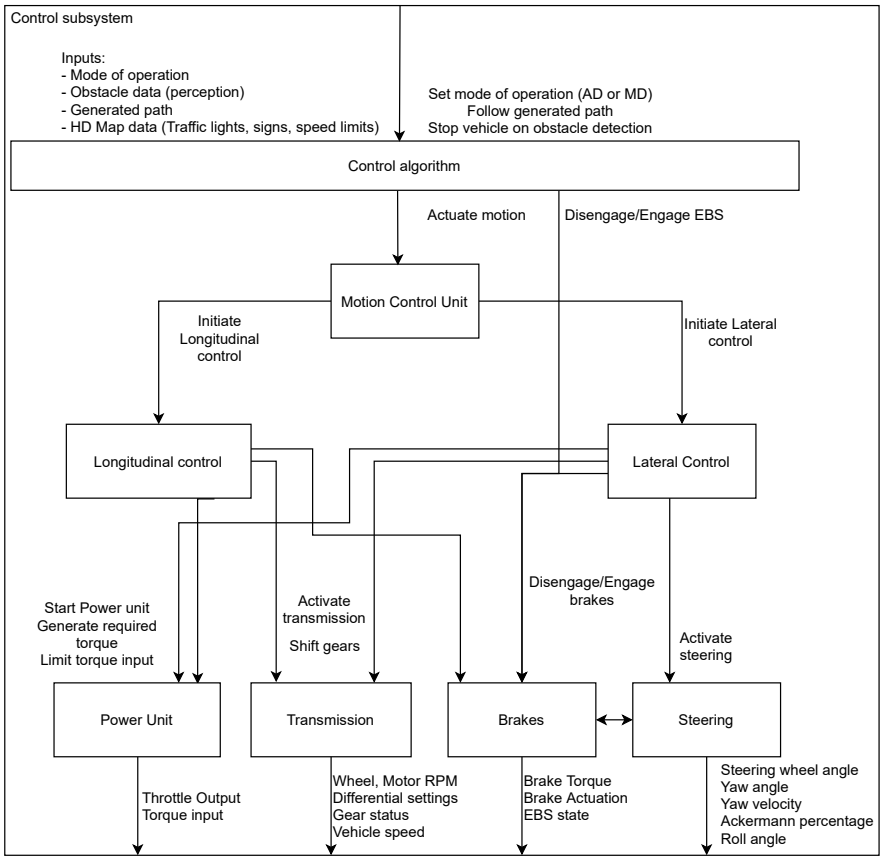


Figure B.4: Control structure of vehicle control system

Figure B.5 describes the path planning and routing subsystems. Using the outputs from the perception, localization and control subsystems, the path planning control unit verifies whether the vehicle is following the defined path. The results of this verification are communicated to the routing subsystem. The routing subsystem requests map data from the data storage unit. The requested data is utilized for analyzing whether the vehicle is still on the right route to the selected destination. If the vehicle strays out of the defined route, the subsystem defines a new route which is communicated to the Path planning control unit. The routing subsystem analyses the detected

obstacles with the detected overhead structures in the map data. Based on the comparison, the autonomous mode can be enabled or disabled. If the disabling instruction is issued, a TOR is issued. Instructions regarding choice of destination and mode of driving issued by the driver from the HMI are communicated to the routing subsystem.

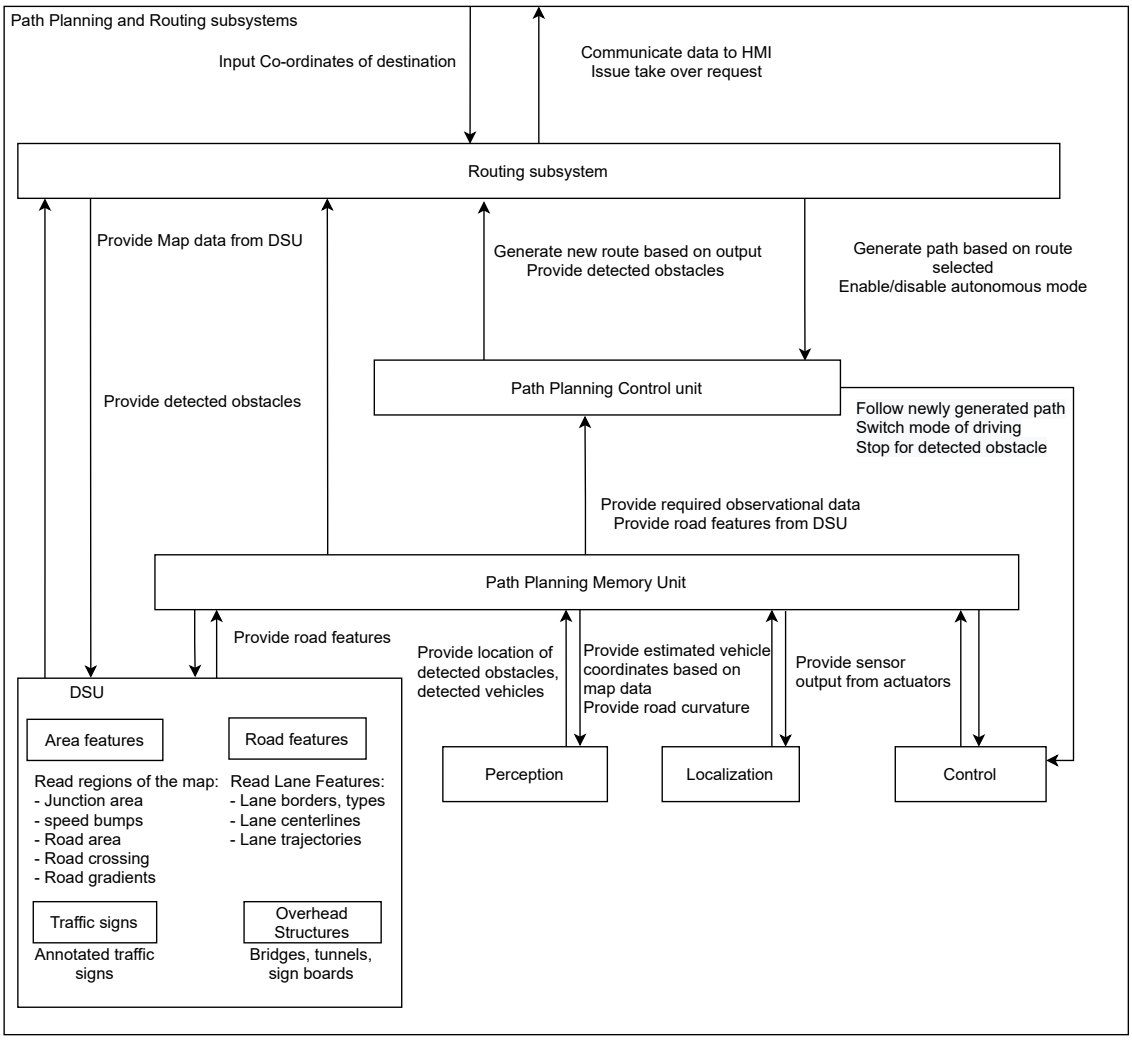


Figure B.5: Control structure of path planning system

## B.3 Loss scenarios

### B.3.1 List of loss scenarios

Table B.2 represents the loss scenarios identified for the control action: Detect traffic signs. Each unsafe control action, which is dependent on map features, has been used in listing out loss scenarios.

Table B.2: List of scenarios for Control Action: Detect traffic signs.

| Unsafe Control action | Scenario classification | Scenario |
|---|---|---|
| The perception algorithm cannot identify traffic signs during autonomous operation. | Inadequate information /feedback is received | When the vehicle is in autonomous mode (AM), the HD map delivery system provides traffic sign features which have incorrect positions. The ASU cannot capture the traffic signs at the defined positions, resulting in no traffic signs being identified. Vehicle intent: Detect and follow traffic signs on the road Role of map: Provide point features for traffic sign detection |
| | Lack of information | When the vehicle is in autonomous mode (AM), the traffic signs have been installed temporarily on a road due to ongoing road works. This does not appear in the point features thus cannot be validated using map data. Vehicle intent: Detect and follow traffic signs on the road Role of map: Provide point features for traffic sign detection |
| | Lack of Information | When the vehicle is operated in autonomous mode(AM), the cameras are unable to capture the traffic signs due to excessive glare (bright sunny day). The perception algorithm must rely on traffic sign features provided in the map. Vehicle intent: Detect and follow traffic signs on the road Role of map: Provide point features for traffic sign detection |
| The perception algorithm incorrectly identifies traffic signs on the road. | Inadequate feedback /information is received | When the vehicle is in autonomous mode (AM), the HD map delivery system provides inaccurate traffic sign features to the perception algorithm for the validation of observational data. This causes incorrect identification of traffic signs on the road. Vehicle intent: Detect and follow traffic signs on the road Role of map: Provide point features for traffic sign detection |
| | Lack of information | When the vehicle is in autonomous mode (AM), the perception algorithm receives map data with insufficient features (missing traffic signs features) for validation of signs detected by the camera. Vehicle intent: Detect and follow traffic signs on the road Role of map: Provide point features for traffic sign detection |
| The perception algorithm identifies traffic signs too late. | Inadequate feedback/ information is received | When the vehicle is in autonomous mode (AM), the ASU detects a traffic light. But the validation of the traffic light with map features requires a longer period due to issues faced in reading map features. Vehicle intent: Detect and follow traffic signs on the road Role of map: Provide point features for traffic sign detection |

### B.3.2 List of high priority loss scenarios

The complete list of loss scenarios obtained after performing the categorisation has been mentioned in this section below. There are 17 loss scenarios which have been labelled as high prioirty. These scenarios have been validated using the sources of use cases.

Table B.3: List of high priority scenarios from STPA.

| Loss scenario ID | Loss scenario Description |
|---|---|
| LS 1 | When the vehicle is operating in autonomous mode (AM) under streetlights or in the foggy conditions, the performance of the camera is hampered. This causes a delay in the lane detection process, which affects the vehicle's localization process.<br>Vehicle intent: Provide vehicle relative location w.r.t lanes to the Path Planning subsystem<br>Role of map: Provide lane borders, trajectories features needed for estimating the vehicle's location w.r.t lanes<br>KPI of features: Positional accuracy (absolute and relative), completeness of lane features |
| LS 2 | When the vehicle is operated in autonomous mode (AM), the overhead structure is out of range for the autonomous sensing unit (ASU), thus cannot be observed by the sensors. The perception algorithm must utilize the overhead structures features to cope with the missing observational data.<br>Vehicle intent: Detect overhead structures/obstacles<br>Role of map: Provide required features needed for identifying overhead structures.<br>KPI of features: Positional accuracy (Absolute), completeness of overhead structures |
| LS 3 | When the vehicle is operating in autonomous mode (AM) in foggy/snowy/heavy rain conditions, the performance of the ASU is heavily hampered. The ASU may not have sufficiently accurate observational data required for detecting overhead structures. The perception algorithm must rely on overhead structures features from the map.<br>Vehicle intent: Detect overhead structures/obstacles<br>Role of map: Provide overhead structures features needed for identifying incoming overhead structures<br>KPI of features: Positional (absolute) accuracy and completeness (false positives and negatives) |
| LS 4 | When the vehicle is operating in autonomous mode (AM) on inner city roads, it encounters missing lane markings. The routing subsystem must combine the vehicle's relative position with the lane dividers and the map's lane centerline information for route generation.<br>Vehicle intent: Use map data for generating a route<br>Role of map: Provide required map data needed for generation of a route<br>KPI of features: Positional (Absolute and relative) accuracy, completeness of lane borders, trajectories |
| LS 5 | When the vehicle is operating in autonomous mode (AM), the HD map delivery system provides a higher (inaccurate) speed restriction. This enables the Path planning Control unit (PPCU) to generate a path requiring a greater speed.<br>Vehicle intent: generate a path using the selected route<br>Role of map: Provide speed restrictions features for path generation<br>KPI of features: Thematic accuracy and completeness (false positive and negatives) of speed restrictions |

Table B.3: List of high priority scenarios from STPA.

| Loss scenario ID | Loss scenario Description |
|---|---|
| LS 6 | When the vehicle is operating in autonomous mode (AM) on a highway, the Path planning Control unit (PPCU) receives inaccurate incoming road curvature information. The PPCU generates a sharper path to be taken by the vehicle.<br>Vehicle intent: generate a path using the selected route<br>Role of map: Provide point features (incoming road curvature) for path generation<br>KPI of features: Heading accuracy and absolute accuracy of curvature of lanes |
| LS 7 | When the vehicle is operating in autonomous mode (AM) in dimly lit conditions, the autonomous sensing unit (ASU) cannot view lane markings with the required confidence levels. Due to low confidence levels of observational data, the perception algorithm cannot detect lane markings.<br>Vehicle intent: Detect and follow lanes on the road<br>Role of map: Provide lane borders, trajectories features for lane detection<br>KPI of features: Positional (Absolute and relative) accuracy, completeness |
| LS 8 | When the vehicle is operating in autonomous mode (AM) during heavy rain, it drives through an underpass where the roads are covered/flooded with water. The flooding covers the lane markings completely, which cannot be identified by the ASU.<br>Vehicle intent: Detect and follow lanes on the road<br>Role of map: Provide lane borders features for lane detection<br>KPI of features: Positional accuracy (absolute), completeness |
| LS 9 | When the vehicle is in autonomous mode (AM), the perception algorithm uses obscured camera data (due to foggy, rainy, snowy conditions) for the detection of lanes with respect to lane features in the map.<br>Vehicle intent: Detect and follow lanes on the road<br>Role of map: Provide lane features required for lane detection<br>KPI of features: Positional, thematic accuracy and completeness of lane features |
| LS 10 | When the vehicle is driving in autonomous mode (AM), the camera encounters lane markings which are difficult to process (yellow colored lane markings) thus forcing the routing subsystem to use line features from the map data for route generation.<br>Vehicle intent: Detect and follow lanes on the road<br>Role of map: Provide lane features required for lane detection<br>KPI of features: Positional, thematic accuracy and completeness |
| LS 11 | When the vehicle is operating in autonomous mode (AM) in heavy rain/snowy/foggy conditions, the autonomous sensing unit (ASU) has limited view of the environment. This results in missing traffic signs on the road and relying on traffic sign features in the map.<br>Vehicle intent: Detect and follow traffic signs on the road<br>Role of map: Provide traffic signs features<br>KPI of features: Coverage, completeness and thematic accuracy of traffic signs |

Table B.3: List of high priority scenarios from STPA.

| Loss scenario ID | Loss scenario Description |
|---|---|
| LS 12 | When the vehicle is in autonomous mode (AM), the traffic signs have been installed temporarily on a road due to ongoing road works. This does not appear in the point features thus cannot be validated using map data.<br>Vehicle intent: Detect and follow traffic signs on the road<br>Role of map:<br>1. Provide point features for traffic signs in areas having sufficient levels of quality<br>2. Provide areas that have lower levels of map quality<br>KPI of features: Completeness and coverage of traffic signs |
| LS 13 | When the vehicle is operated in autonomous mode (AM), the cameras are unable to capture the traffic signs due to excessive glare (bright sunny day). The perception algorithm must rely on traffic sign features provided in the map.<br>Vehicle intent: Detect and follow traffic signs on the road<br>Role of map: Provide point features for traffic sign detection<br>KPI of features: Coverage and Thematic accuracy of traffic signs |
| LS 14 | When the vehicle is in autonomous mode (AM), the traffic sign is located in a blind spot due to surrounding traffic. The autonomous sensing unit (ASU) cannot detect the traffic sign, thus requiring the traffic sign features from the map.<br>Vehicle intent: Detect and follow traffic signs on the road<br>Role of map: Provide point features for traffic sign detection<br>KPI of features: Completeness and thematic accuracy of traffic signs |
| LS 15.1 | When the vehicle is operating in autonomous mode (AM), the traffic light is out of range of detection for the autonomous sensing unit (ASU). The vehicle stops at an angle w.r.t the traffic light, which makes its hard to detect.<br>Vehicle intent: Detect and follow traffic lights on the road<br>Role of map: Provide traffic lights information (Geometrical information of the bounding box)<br>KPI of features: Positional accuracy of traffic lights |
| LS 15.2 | When the vehicle is operating in autonomous mode (AM), the traffic light is out of range of detection for the autonomous sensing unit (ASU). The vehicle stops in front of the traffic light, which makes it hard to detect the traffic light. The vehicle must move backward to get into position to detect the traffic light appropriately.<br>Vehicle intent: Detect and follow traffic lights on the road<br>Role of map: Provide traffic lights information (Geometrical information of the bounding box)<br>KPI of features: Positional accuracy of traffic lights |

Table B.3: List of high priority scenarios from STPA.

| Loss scenario ID | Loss scenario Description |
|---|---|
| LS 16 | When the vehicle is operating in autonomous mode (AM), the routing subsystem generates a route for the vehicle to follow in an inner city environment. However, after this point, the autonomous sensing unit (ASU) pinpoints the vehicle's current position inaccurately due to multiple buildings on each side of the road.<br>Vehicle intent: Use map data for generating a route<br>Role of map: Provide required map data needed for generation of a route<br>KPI of features: Positional accuracy of traffic signs |

### B.3.3 Root-cause analysis of high priority scenarios

In this section, a root-cause analysis of the scenarios has been conducted . Each scenario has been split into two main categories: unforeseeable and foreseeable [50]. Furthermore, the type of entities present in the scenario along with the event occurring in the scenario has been listed [50]. This has been performed for all the high priority scenarios mentioned in the previous subsection.

Table B.4: Root-cause analysis of high-priority scenarios.

| **Loss scenario ID** | **Unforesee -able** | **Foresee-able** | **Dynamic Entities** | **Static Entities** | **Events** |
|---|---|---|---|---|---|
| LS 1 | NA | Preventable | Road features | Vehicle, traffic | Foggy conditions → camera performance hindered |
| LS 2 | NA | Preventable | Overhead structures, Road features | Vehicle, traffic | Failure to identify overhead structures |
| LS 3 | NA | Unpreventable | Road features, Overhead structures | Vehicle, traffic | Foggy conditions → camera performance hindered |
| LS 4 | NA | Unpreventable | Road features (lane markings) | Vehicle, traffic | Missing lane markings → failure to identify lanes |
| LS 5 | NA | Preventable | Speed limit signs, Road features | Vehicle, traffic | Inability to read speed limit signs → Incorrect speed limit received from map |
| LS 6 | NA | Unpreventable | Road features, (curvature signs) | Vehicle, traffic | Incorrect road curvature information received from map |
| LS 7 | NA | Preventable | Road features (lane markings) | Vehicle, traffic | Poor lighting conditions → inability to observe lane markings |

Table B.4: Root-cause analysis of high-priority scenarios.

| **Loss scenario ID** | **Unforesee -able** | **Foresee-able** | **Dynamic Entities** | **Static Entities** | **Events** |
|---|---|---|---|---|---|
| LS 8 | NA | Unpreventable | Road features (lane markings) | Vehicle, traffic | Heavy rain inside an underpass → Flooded lane markings |
| LS 9 | NA | Unpreventable | Road features (lane markings) | Vehicle, traffic | Weather conditions → Inability to observe and detect lanes |
| LS 10 | NA | Unpreventable | Road features (Yellow lane markings) | Vehicle, traffic | Yellow lane markings → Failure of detection algorithm |
| LS 11 | NA | Unpreventable | Road features, traffic signs | Vehicle, traffic | Weather conditions → Failure in observing and detection of traffic signs |
| LS 12 | Yes | NA | Road features, traffic signs, construction elements | Vehicle, traffic, construction workers | Inability to detect construction signs/ boards → Continued movement of vehicle |
| LS 13 | NA | Unpreventable | Road features, traffic signs | Vehicle, traffic, pedestrians | Weather conditions → Compromising of performance of cameras |
| LS 14 | NA | Unpreventable | Road features, traffic signs | Vehicle, traffic | Larger vehicle occupies neighboring lane → Blocking of FOV of vehicle's camera |
| LS 15.1 | NA | Preventable | Road features, traffic lights | Vehicle, traffic lights, pedestrians | Vehicle is positioned at an angle → Delay in traffic light estimation |
| LS 15.2 | NA | Preventable | Road features, traffic lights | Vehicle, traffic lights, pedestrians | Vehicle stops at traffic light below the traffic light → Delay in traffic light estimation |
| LS 16 | NA | Unpreventable | Buildings, Road features, traffic signs | Vehicle, traffic lights, pedestrians | Vehicle enters a region with multiple buildings → GPS is compromised |

## B.4 Scenario validation

### B.4.1 Process for scenario validation

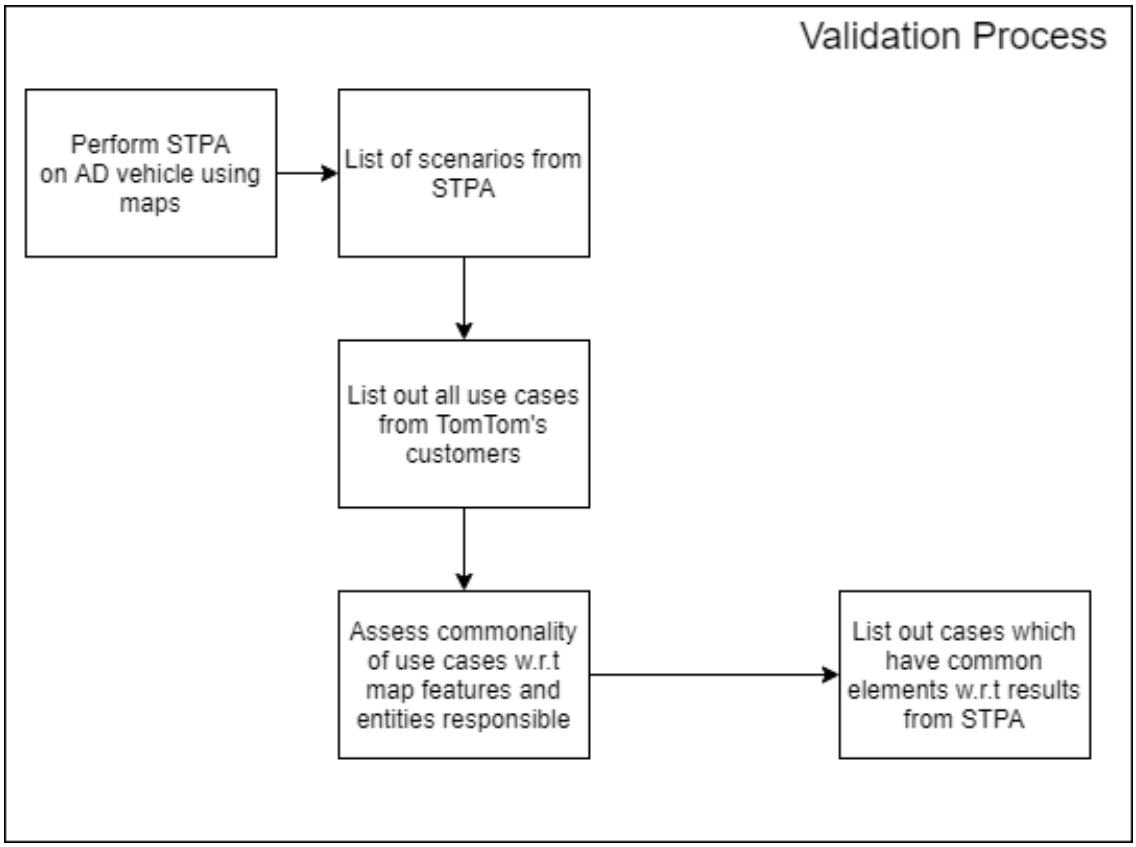


Figure B.6: Process for validation of scenarios.

### B.4.2 List of scenarios identified by TomTom's clients

Table B.5: List of scenarios identified by TomTom's clients.

| Scenario ID | Scenario description |
|---|---|
| AP1 | The vehicle receives an incorrect speed limit from the map data. |
| AP2 | The vehicle receives inaccurate lane features such as lane width and lane curvature. |
| AP3 | The vehicle receives a map with lane features containing lateral inaccuracies or disjointed curves |
| AP4 | The vehicle drives in autonomous mode during lightning or heavy rain conditions resulting in the failure of the lane detection system. The vehicle uses map data that contains lateral inaccuracies. |
| AP5 | The vehicle receives inaccurate area features from the map concerning a toll booth area, resulting in no take over being issued by the system. |
| AP6 | A construction zone in bright stadium lighting/floodlights or at sunrise or in snowy conditions cannot be detected by the vehicle |
| AP7.1 | The vehicle drives into an underpass or an overhead structure with artistic lighting (unconventional lighting patterns). |

Table B.5: List of scenarios identified by TomTom's clients.

| Scenario ID | Scenario description |
|---|---|
| AP7.2 | A. The vehicle approaches an underpass which is flooded due to incessant rainfall<br>B. The vehicle hydroplanes due to excessive waterlogged on the road |
| AP8 | The vehicle drives on a road with missing lane markings on the road edge/ lane markings are non-existent. |
| AP9 | The vehicle approaches a service road which is present laterally next to the highway lane. |
| AP10 | There are multiple barriers placed one behind the other at the sides of the road. The vehicle fails to detect the second barrier placed. |
| AP11 | The vehicle approaches a toll booth area which at the exit, has missing lane markings. |
| AP12 | The vehicle drives into a single-lane tunnel but the map detects an incoming toll booth . |
| AP13 | The vehicle approaches an area with stacked flyovers/highways. |

### B.4.3 List of scenarios identified using TomTom's Measurement data

Table B.6: List of scenarios identified using TomTom's measurement data.

| Scenario ID | Scenario description |
|---|---|
| MM1 | The vehicle's camera is obstructed from observing/detecting traffic signs by vehicles in the neighboring lane |
| MM2 | The vehicle cannot detect lanes on exiting a road below an overhead structure due to sudden change in brightness |
| MM3 | The vehicle is unable to detect lanes under conditions of heavy rain. |
| MM4 | The vehicle encounters a false positive whilst detecting lanes resulting in two sets of lanes detected on the same side |
| MM5 | The vehicle encounters an area with missing or faded/dull lane markings, which results in no lanes being detected by the Lane detection system. |
| MM6 | The vehicle's lane detection system produces irregular output whilst driving on a straight road. |
| MM7 | When the vehicle exits the highway, it encounters a splitting of lanes which are detected as two sets of lanes. |
| MM8 | The vehicle cannot detect traffic signs under sunny conditions |
| MM9 | The vehicle's location estimation deviates when it is at the below ground level on exiting an underground tunnel/highest elevation of overhead structure. |
| MM10 | When the vehicle is driving in a tunnel, the lane detection system works ineffectively due to insufficient lighting or yellow lighting. |
| MM11 | When the vehicle is driving on an overhead structure, the lateral accuracy of the vehicle's location estimate decreases. |
| MM12 | The vehicle cannot detect a traffic sign due to graffitti art on it. (False negative) |
| MM13 | The vehicle detects a false positive traffic sign whilst driving behind a truck. |
| MM14 | When the vehicle is driving in a tunnel, it cannot stream lane border information correctly resulting in an incorrect overlay with the detected lane markings |
| MM15 | Lane border information streamed by the vehicle is at an offset to the lane markings on the road |

Table B.6: List of scenarios identified using TomTom's measurement data.

| **Scenario ID** | **Scenario description** |
|---|---|
| MM16 | The vehicle streams lane border information incorrectly, resulting in lane borders matching with lane centerlines on the road |
| MM17 | The vehicle streams traffic sign information which contains false positives (no traffic sign present at the defined location) |
| MM18 | The vehicle streams traffic sign information with the bounding box placed incorrectly over the traffic signs mounted on the road. |
| MM19 | The vehicle streams traffic sign information with missing traffic sign information in it. |

# Appendix C

# Python Code

## C.1 Functions for Pure Pursuit control

### C.1.1 Estimation of goal point using defined look-ahead distance

```
#function for generating the next waypoint vehicle must follow
def get_next_waypoint(world, vehicle, waypoints):
    vehicle_location = vehicle.get_location()
    min_distance = 50 #ideally set it to 200 for testing throttle = 0.5, 300
    next_waypoint = None

    for waypoint in waypoints:
        waypoint_location = waypoint.transform.location

        #Only check waypoints that are in the front of the vehicle (if x is
            negative, then the waypoint is to the rear)
        #TODO: Check if this applies for all maps
        if (waypoint_location - vehicle_location).x > 0 or (waypoint_location -
            vehicle_location).y < 0 :
            #Find the waypoint closest to the vehicle, but once vehicle is close to
                upcoming waypoint, search for next one
            if vehicle_location.distance(waypoint_location) < min_distance and
                vehicle_location.distance(waypoint_location) > 0.05: #default was 5
                metres
                min_distance = vehicle_location.distance(waypoint_location)
                next_waypoint = waypoint
    return next_waypoint
```

### C.1.2 Implementation of Pure pursuit controller

```
# controller.py
import glob
import os
import sys


try:
    sys.path.append(glob.glob('../carla/dist/carla-*%d.%d-%s.egg' % (
        sys.version_info.major,
        sys.version_info.minor,
        'win-amd64' if os.name == 'nt' else 'linux-x86_64'))[0])
except IndexError:
    pass

import carla
import numpy.matlib
```

```
import numpy as np
import math

#----define a controller for driving the vehicle on the defined path----#
def control_pure_pursuit(vehicle_tr, waypoint_tr, max_steer, wheelbase):
    # TODO: convert vehicle transform to rear axle transform
    wp_loc_rel = relative_location(vehicle_tr, waypoint_tr.location) + carla.
        Vector3D(wheelbase, 0, 0)
    wp_ar = [wp_loc_rel.x, wp_loc_rel.y]
    d2 = wp_ar[0]**2 + wp_ar[1]**2
    #steer_rad = math.atan(2 * wheelbase * wp_loc_rel.y / d2)
    steer_rad = math.atan2(2 * wheelbase * wp_loc_rel.y, d2)
    steer_deg = math.degrees(steer_rad)
    steer_deg = np.clip(steer_deg, -max_steer, max_steer)
    return steer_deg / max_steer

def relative_location(frame, location):
  origin = frame.location
  forward = frame.get_forward_vector()
  right = frame.get_right_vector()
  up = frame.get_up_vector()
  disp = location - origin
  x = np.dot([disp.x, disp.y, disp.z], [forward.x, forward.y, forward.z])
  y = np.dot([disp.x, disp.y, disp.z], [right.x, right.y, right.z])
  z = np.dot([disp.x, disp.y, disp.z], [up.x, up.y, up.z])
  return carla.Vector3D(x, y, z)
```

## C.2 Noise injection in waypoints

```
import glob
import os
import sys

try:
    sys.path.append(glob.glob('../carla/dist/carla-*%d.%d-%s.egg' % (
        sys.version_info.major,
        sys.version_info.minor,
        'win-amd64' if os.name == 'nt' else 'linux-x86_64'))[0])
except IndexError:
    pass

import carla
import numpy.matlib
import numpy as np
import math

def noise_add(noise_x,noise_y,noise_z,waypt):
    rotation = carla.Rotation(0,0,0)
    location = carla.Vector3D(0,0,0)
    wp = carla.Transform(location, rotation)

    wp.location.x = waypt.transform.location.x + noise_x
    wp.location.y = waypt.transform.location.y + noise_y
    wp.location.z = waypt.transform.location.z + noise_z
    return wp
```