CRANFIELD UNIVERSITY

Marc Martínez Jiménez

Development of an ADAS function for combined adaptive cruise control and lane keeping assist up to highway speeds

School of Aerospace, Transport and Manufacturing (SATM)
Automotive Mechatronics MSc

Academic Year: 2020 - 2021

Supervisor: Efstathios Velenis
Associate Supervisor: Efstathios Siampis, Clara Marina Martínez
August 2021

ABSTRACT

Due to the need of higher control strategies for the viability of autonomous vehicles on an ever changing environment, this thesis will cover the development of an autonomous vehicle simulation considering longitudinal control from an Adaptive Cruise Controller (ACC) and lateral control from a Lane Keeping Assist (LKA) controller. The implementation of the control model and code will be made in Simulink and Matlab and simulated using the virtual testing software IPG CarMaker.

The vehicle simulated is a Volkswagen Beetle equipped with two sensors. For the ACC, a radar sensor is placed at the front bumper providing relevant information about the preceding traffic object, in particular, the presence of an object and the relative distance and velocity. On the other hand, a line sensor is placed behind the windshield rear-view mirror to monitor the lane bounds and provide the relative position of the lane with respect to the sensor. A line sensor, essentially, emulates a camera sensor with some image processing layers capable of reliably detect lines.

To enable longitudinal control, the Intelligent Driver Model calculates the desired acceleration by comparing the desired ego car speed with the actual value. Also, it compares the actual relative distance to a desired distance calculated by considering a standstill gap, a desired time gap at a certain speed and the relative velocity with a preceding vehicle. This model though does not precisely track the relative distance when approaching the desired speed. To address this issue, an algorithm that enables this functionality is implemented monitoring the relative velocity beyond the speed limit to avoid undesired driving conditions.

On the other hand, lateral control is achieved by considering both degrees of freedom of a four wheeled vehicle, yaw angle and lateral deviation from the lane centreline. For this, Catmull-Rom curves are fitted in the lane centre to measure both magnitudes and compare them to their ego vehicle's respective values. Then, both errors are fed to their respective PID controllers and combined into one signal to command the steering angle of the vehicle.

Combining both functionalities results in loss of acceleration control for approaching curves and, thus, a method for such task is developed. The algorithm consists in monitoring the curvature of a moving point on the fitted curves and modify the desired speed inversely proportional to it.

Finally, a safety measure is implemented for instances where a preceding vehicle disappears from the radar sensor scope. By monitoring the intersection of the path with the sensor cone, the ego vehicle expects a vehicle to enter the radar scope through that point and adapts the acceleration accordingly.

To evaluate the performance of the controller, several test cases are executed both assessing the ACC and LKA capabilities by its own, and the combined implementation. For this, synthetic and real life scenarios are tested with and without traffic.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF APPENDIX FIGURES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

ACC	Adaptive Cruise Control
LKAS	Lane Keeping Assist System
LKA	Lane Keeping Assist
IDM	Intelligent Driver Model
COG	Centre Of Gravity

1 Introduction

As the prevalence of intelligent vehicles is increasing along with their relevance in the automotive markets, the necessity of fully autonomous vehicles is a requirement for their future success. To reach the goal of fully autonomous vehicles, first, several milestones of increasing complexity need to be accomplished. These milestones are categorised into different levels according to the Society of Automotive Engineers (SAE) ranging from 0 to 5, where 0 means the vehicle is fully controlled by the driver and only gives safety warnings, and 5 means the driver does not need to involved in driving in any way.

In this thesis, the focus will be centred in the automating capabilities of level 2 and level 3. Level 2 means that driving actions such as accelerating, braking and steering are controlled by the vehicle, but the driver must monitor the vehicle driving and take control whenever necessary. One step ahead in automation is level 3, where the driver no longer needs to be constantly monitoring the vehicle's behaviour on the road but it is required to take control of the driving as soon as the vehicle requires it, usually in dangerous or in not considered situations. Nowadays, the majority of automated systems in the market lay in level 2, as the step to level 3 usually requires a lot of precise monitoring of the environment which can be technically difficult to implement.

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	**High Automation**	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 1-1: SAE levels and brief description of its ranges

The work done on this project will consider an ACC strategy to control the longitudinal behaviour, mainly acceleration and braking, and a LKA control strategy to modify the lateral manoeuvres of the vehicle, in particular, the steering angle and velocity. Thus, both functionalities will ensure automated capabilities up to level 2.

To do so, a model for both functionalities will be developed from in Matlab/Simulink and the algorithm will be tested in the simulation software IPG CarMaker, that considers vehicle dynamics and all the related physics of automotive vehicles.

1.1 Aim and objectives

This project aims at developing an ADAS functionality consisting on adaptive cruise control and lane keeping assist for longitudinal and lateral control, respectively. To do so, first, a thorough understanding of the software needs to be reached, mainly Matlab/Simulink and IPG Carmaker. Also, research needs to be conducted to gain insight on the sensors used and their placement inside the vehicle. Knowledge about the currently used controllers for autonomous vehicles will also be sought after, as it can provide a strong basis from where to kick-start the project.

Once all prior knowledge of the topic is gathered, the two functionalities will be implemented separately focusing on reaching at least Level 2 autonomy for each. When both implementations have reached Level 2, they will be combined to get the full implementation and see the cross coupling effects each functionality has on the other. Finally, if the combination is successful, further features will be added to attempt reaching Level 3 and increase vehicle safety and autonomy.

If the development of the combined implementation is satisfactory, several test cases will be proposed in order to assess not only the combined controller, but also each feature by its own.

2 Literature Review

2.1 ADAS virtual validation: ACC and AEB case study with IPG CarMaker [1]

This master's thesis mainly covers the development of an adaptive cruise control (ACC) and adaptive emergency braking (AEB) functionality using IPG Carmaker as the automotive simulator. For this, this thesis goes deeply into the CarMaker model developed and the Matlab/Simulink controllers devised.

First, a thorough description of the CarMaker model is provided. This model consists of taking the default car native to CarMaker and modify it to accept acceleration commands from Simulink. To do this, an object sensor is placed in the front bumper of the car to simulate the detections and relevant magnitudes provided by a radar sensor with some data processing. Also, a vehicle controller for longitudinal acceleration control needs to be added in Carmaker's model to grant Direct Variable Access (DVA) to the Simulink commands generated.

Then, a Simulink breakdown of the acceleration controller is given. In particular, the ACC and AEB controllers consist in measuring the relative distance and speed of the preceding vehicle with respect to the ego vehicle in conjunction with the absolute velocity of the ego car. Then, it compares each one to its desired value and multiplies the resulting error by a manually preselected constant. Depending on the scenario the car is in, a switch chooses which error considers for the acceleration calculation. The error chosen can either be the combination of the relative speed and distance if there is a preceding car in sight, or the ego car's velocity error if there is none. In broad terms, this controller works as a PD controller on the distance error when there is a preceding vehicle ahead of the ego vehicle, as it considers the relative distance and its rate of change, and a proportional speed controller when there is no vehicle in the sensor scope.

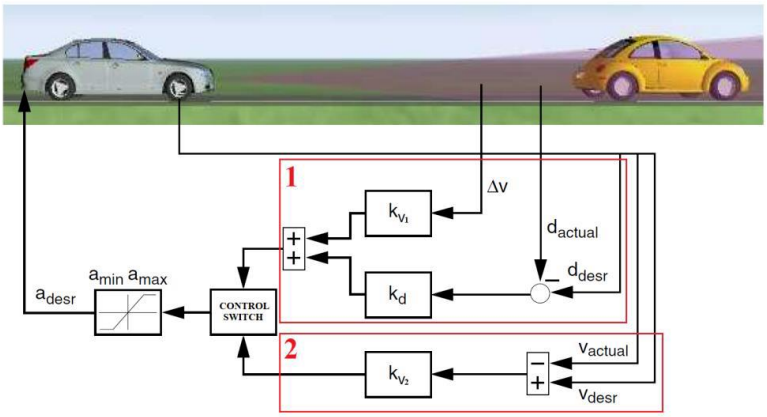

Figure 2-1: ACC and AEB controller developed [1]

Finally, to test the validity of the functionality and how it relates to the real world, several test cases are run, always trying to find the edge cases of the algorithm. For this project, what is of most interest is the development of the ACC functionality and the test cases considered for a proper testing strategy of the algorithm. As seen in Figure 2-2, these test scenarios cover different traffic configurations that test all the possible situations where a preceding vehicle could interfere with the ego vehicle.

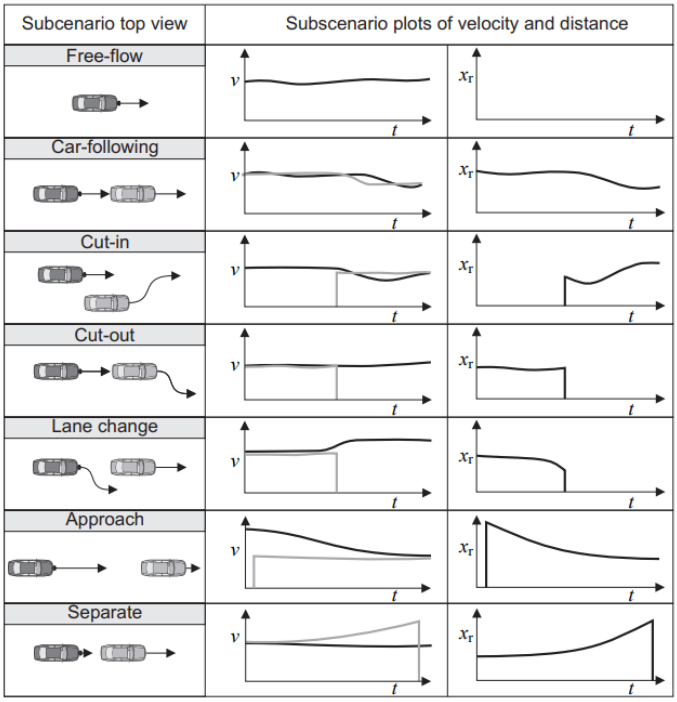

Figure 2-2: Typical test cases for an adaptive cruise controller [2]

2.2 Adaptive cruise control design for active congestion avoidance [3]

As the title suggests, this paper covers an approach on to how to adapt the acceleration and braking strategies depending on the traffic situation. For this, the traffic scenarios are stratified into five levels (*Free traffic, Upstream jam front, Congested traffic, Downstream jam front, Bottleneck sections*) and, depending on the congestion level, the acceleration/braking behaviour is adapted.

To implement this functionality, the intelligent driver model is presented and used to control not only the acceleration of the ego car, but also the traffic vehicles. This model has the following expression:

$$\dot{v}(s, v, \Delta v) = a \cdot \left[1 - \left(\frac{v}{v_0}\right)^4 - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right] \quad \text{(Eq 2-1)}$$

$$s^*(v, \Delta v) = s_0 + v \cdot T + \frac{v \cdot \Delta v}{2 \cdot \sqrt{a \cdot b}} \quad \text{(Eq 2-2)}$$

Where s and Δv are the relative distance and velocity between the preceding car and the ego vehicle, respectively; v is the absolute velocity of the ego car; a is the maximum desired acceleration possible, v_0 is the desired speed, for example, the maximum road speed; s_0 is the standstill safety distance; T is the desired time gap between both vehicles; and b is the desired/comfortable deceleration.

This equation approaches the adaptive cruise control strategy in a continuous manner and, thus, no switching strategy needs to be implemented which is easier to implement and makes the system less complex. This strategy will be the base from where the longitudinal control of this project will be built. As it can be seen, this controller loses distance tracking capabilities whenever the ego vehicle reaches the desired speed. This comes from the fact that the speed ratio $\left(\frac{v}{v_0}\right)^4$ becomes closer to one and, thus, the distance ratio needs to decrease in order to achieve steady state, $\dot{v}(s, v, \Delta v) = 0$. Close to steady state conditions and at large speeds, s* is mainly composed by the time gap term and, thus, nearly constant. As a consequence, to reduce the distance ratio $\left(\frac{s^*(v,\Delta v)}{s}\right)^2$ the measured distance need to increase, loosing distance tracking capabilities. This implies that the distance between the preceding and the ego vehicle needs to increase beyond the desired distance in order to accomplish steady state conditions,

which is not desirable. The major improvement of this project will be to address this issue safely and respecting the road laws.

2.3 Real Time Lane Detection and Tracking System Evaluated in a Hardware-in-The-Loop Simulator [4]

This paper covers the pipeline followed to achieve good lane marking detections and proper processing of the data to, in the end, control the steering wheel to keep the ego vehicle between the lane markings. Also, it describes procedures to make the lane detection search much more efficient and, hence, being capable of ensuring that the algorithm runs in real time.

Concerning the lane detection pipeline, first, the bird's eye point of view from a is extracted by applying a projective transform to the image pixels. In broad terms, this procedure consists in transforming the converging perspective lines of an image to being parallel and vertical. If done properly, this conversion achieves lanes and lane marking of constant width, thus, improving lane detection.

Further down the pipeline, a Gabor filter is applied to highlight the lane edges and extract the intensity histogram of the pixels. With these histograms, a window/region of interest is drawn around them to make the lane search more efficient. To find the top coordinates of the lane markings, the search is started from the top left corner of the windows, while for the bottom points the search starts at the bottom left corner. Once the initial and final points are found, a middle point is found by starting the search from the average longitudinal and lateral position of the two points previously detected. With these points, a Bezier cubic curve is fitted. To do so, control points need to be found so that the resulting Bezier best matches the points found using the lane detection markings.

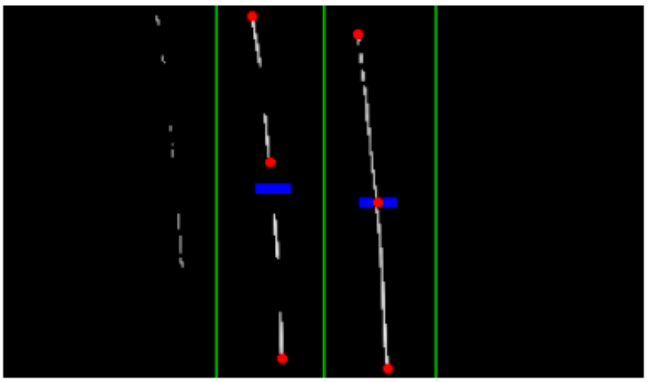

Figure 2-3: Search window (green); Points detected (red); Middle search start point (blue)

Also, once the curves are fitted, a region surrounding the Bezier curves can be defined so that the following lane search is begun in this region, minimising the search time. Finally, the paper states that in order to control the steering commands, a simple PD controller taking into account only the lateral deviation of the sensor to the lane markings is enough to properly follow the lane.

2.4 A path-following driver model with longitudinal and lateral control of vehicle's motion [5]

This paper develops a longitudinal and lateral control strategy to adapt both the steering angle and the speed based on information provided by the path ahead.

For longitudinal control, the curvature of the previewed path is monitored on a moving point along it. This curvature directly influences the maximum desired speed by considering the following expressions:

$$a_y = \frac{U^2}{R} = k_t \cdot U^2 \qquad \text{(Eq 2-3)}$$

Where a_y is the lateral acceleration of the vehicle during certain curvature manoeuvres; U is the longitudinal vehicle speed; R is the curve turning radius; and k_t is the curve's curvature, the radius inverse. Thus:

$$U_{max} = \sqrt{\frac{a_{y,max} \cdot \mu_y}{|k_t|}} \qquad \text{(Eq 2-3)}$$

Where $a_{y,max}$ is the vehicle's characteristic maximum lateral acceleration, μ_y is the lateral coefficient of friction; and U_{max} is the curve's maximum speed required with a given

maximum lateral acceleration and curvature. Notice that the curvature in this equation is passed through an absolute value operator to avoid singular mathematical cases, like, in this case, negative values inside a square root operator. This concept can also be understood as the maximum velocity not being affected by the direction of the curve, from where the curvature sign convention comes from.

The moving point distance to the vehicle is calculated proportional to the vehicle's squared velocity and inversely proportional to the maximum braking deceleration $a_{b,max}$ and the tyre-road longitudinal coefficient of friction μ_x.

$$s_{p,1} = \frac{U^2}{2 \cdot a_{b,max} \cdot \mu_x} \quad \text{(Eq 2-4)}$$

By doing so, the vehicle will monitor the curvature of a farther point when developing high speeds, hence, giving more head time to react, for instance, in case of a high curvature curve at high speeds. This, in turn, lets the longitudinal controller theoretically slow down the vehicle in a smother way, improving passenger comfort and safety.

Finally, for longitudinal control, the maximum speed from a specific curve is compared with the actual speed and the error is passed through a proportional controller with constant K_U. The output e_U of this controller will command the acceleration/brake approach of the gas/brake pedals.

$$e_U = K_U \cdot (U_{max} - U) \quad \text{(Eq 2-5)}$$

For lateral control, both yaw and lateral deviation are considered to control the steering angle. For yaw, the heading angle of the path is measured at a speed dependent moving point of the previewed path.

$$s_{p,s} = U \cdot T_p \quad \text{(Eq 2-6)}$$

Where T_p is a user selected parameter to control how far ahead the preview point is at a certain speed. Then, the error is calculated, in broad terms, by comparing it with the vehicle yaw angle and fed to a PD controller to generate the first control action.

$$\delta_\psi = K_{\psi,P} \cdot e_\psi + K_{\psi,D} \cdot \frac{de_\psi}{dt} \quad \text{(Eq 2-7)}$$

On the other hand, the lateral deviation error is calculated by comparing the lateral position difference between the same previewed point of the path with the car position.

This deviation error can also be calculated by measuring the relative position of the moving point with respect to the ego vehicle. This error, though, is passed through a proportional gain to achieve the second steering angle signal.

$$\delta = K_{d,P} \cdot e_d \quad \text{(Eq 2-8)}$$

Finally, both signals are condensed into one with a gain scheduling strategy that makes the steering commands less responsive as the longitudinal speed increases.

$$\delta = \frac{1}{K_{rd}} \cdot (\delta_\psi + \delta_d) \quad \text{(Eq 2-9)}$$

Where δ is the desired steering angle and K_{rd} is the gain scheduling constant.

$$K_{rd} = \frac{r}{\delta} = \frac{U}{l \cdot (1 + K \cdot U^2)} \quad \text{(Eq 2-10)}$$

Where l is the vehicle wheelbase and K is the understeering constant.

Finally, this combined longitudinal and lateral controller is only tested to be able to reliably follow a double lane change trajectory.

3 Methodology

3.1 Software overview

This project is mainly developed using Matlab/Simulink for the controller implementation, mainly, coding and modelling. IPG CarMaker is used for testing the algorithm on different simulated driving scenarios.

3.1.1 Matlab/Simulink

Matlab is a programming and computing platform used for data analysis, algorithm development and model creation. [6] During the development of this project, Matlab has been mainly used for data analysis of information extracted from Simulink and code development for difficult to implement algorithms on Simulink.

Simulink is a block based platform for simulation and model design/creation. Particularly, the main advantage of this platform in front of Matlab is that the simulation software IPG CarMaker has already some examples implemented in Simulink that simultaneously run CarMaker. This means that for controller development, focus only needs to be poured into reading the relevant/desired information from the sensors placed on the CarMaker vehicle, developing proper controllers and adapting the generated signals to the simulator's requirements. Due to this, no attention needs to be directed into developing the simulator itself as it is already managed by the already existing interface between Simulink and CarMaker.

Figure 3-1: IPG Carmaker for Simulink interface

3.1.2 IPG CarMaker

On the other hand, IPG CarMaker is a software made specifically to accurately simulate automotive physics of different vehicles, in various test scenarios with changing environments. This simulator also provides visualization and plotting features that make software development much more streamlined and optimised.

For it to properly simulate the desired behaviours, a Car model needs to be specified, either from the pre-existing examples or a new custom one. Also, extra parameters, such as the vehicle tire specifications, a carried trailer or extra vehicle load can be specified in case the automatically generated parameters do not fit the desired ones.

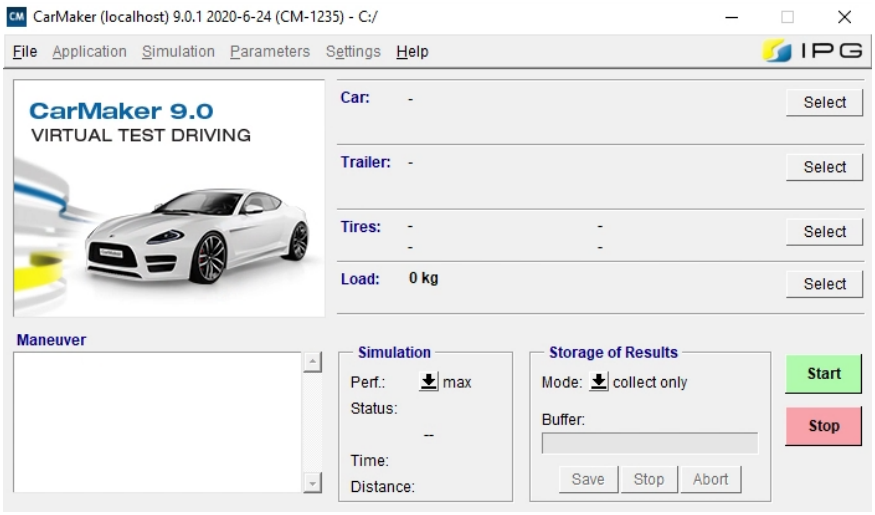

Figure 3-2: IPG CarMaker GUI

In addition to the vehicle related options, a road manoeuvre is needed so that the vehicle knows what are the desired actions it needs to take for a particular test case. Manoeuvres comprise of a wide variety of driving actions both lateral and longitudinal, such as lane changes, acceleration specification, autonomous driving, etc.

Finally, to run a basic test, a road scenario has to be specified in order to locate the vehicle in some road profile throughout the test case. This scenario can be synthetically generated or imported from external sources such as Google Maps to emulate various real road scenarios.

In this thesis, the specified longitudinal manoeuvres will be overwritten by the acceleration control commanded by the adaptive cruise control. Also, the steering wheel angle and velocity will be overtaken by the lane keeping assist. Finally, there will be traffic added to test the different capabilities of both controllers developed.

3.2 Controller development

3.2.1 Adaptive Cruise Control

As previously stated, the acceleration control is based on the use of the Intelligent Driver Model (IDM) presented on [3]. This model has the following equations:

$$\dot{v}(s, v, \Delta v) = a \cdot \left[1 - \left(\frac{v}{v_0}\right)^4 - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right] \quad \text{(Eq 2-1)}$$

$$s^*(v, \Delta v) = s_0 + v \cdot T + \frac{v \cdot \Delta v}{2 \cdot \sqrt{a \cdot b}} \quad \text{(Eq 2-2)}$$

Where s and Δv are the relative distance and velocity, respectively, between the preceding car and the ego vehicle; v is the absolute velocity of the ego vehicle; a is the maximum acceleration desired, v_0 is the desired speed, for example, the maximum road speed; s_0 is the standstill safety distance; T is the desired time gap between both vehicles; and b is the desired/comfortable deceleration.

In particular, this approach to ACC can be understood differently whether or not a car is in the scope of the ACC radar sensor. If there is not a vehicle in the scope, s is assumed to be large making the distance ratio negligible, and the car will accelerate until the actual velocity matches the desired one. This part can be expressed with only the first two terms as:

$$\dot{v}(s, v, \Delta v) = a \cdot \left[1 - \left(\frac{v}{v_0}\right)^4\right] \quad \text{(Eq 3-1)}$$

On the other hand, when there is a traffic object in the radar scope, the acceleration change provided by distance variations is more significant if the ego vehicle's speed is relatively smaller than the desired one. This is due to the fact that the distance ratio is raised to the power of two and the velocity ratio is to the power of four. Since the ratios are usually lower than one in value, the square of such number is way bigger than a ratio raised to the power of four, as in the velocity ratio.

In addition, the distance ratio considers different scenarios when a preceding vehicle is in the sensor scope. The standstill distance s_0 achieves a safety distance when velocity is zero. The desired time gap T manages to keep a safe distance at constant velocity, for instance, in car following scenarios. Finally, the relative velocity Δv controls the changes of velocity in order to brake/accelerate in transient acceleration scenarios

between steady state conditions, for instance, emergency braking or approach/separate scenarios.

The sign of the relative velocity for the IDM, though, is defined as the opposite of the intuitive value. Usually, the sign of the relative velocity states that a reduction of the relative distance implies a negative relative velocity, that is, when the vehicle in front is suddenly braking or the ego car is approaching the preceding car. Similarly, when the preceding vehicle is driving away or the ego car is increasing the distance with the preceding car and, hence, the relative distance is increasing, the intuitive relative velocity sign is positive. The IDM's Δv has the opposite behaviour. Increasing the relative distance implies a negative relative velocity and a decrease of it implies a positive relative velocity. This is done so that the desired distance with the preceding car, s*(v, Δv), increases when the relative distance is decreasing, braking consequently and vice versa.

This algorithm is implemented in Simulink as follows:

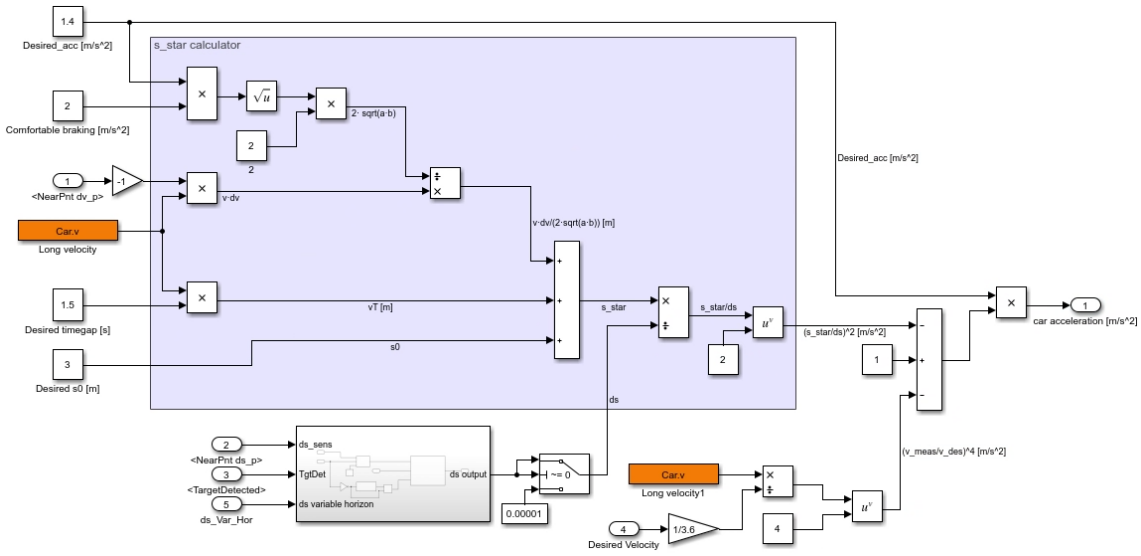

Figure 3-3: Matlab/Simulink diagram for the IDM implementation

As it can be seen, the relative velocity from CarMaker has to be multiplied by a -1 gain to address the change of sign that the IDM requires to properly function.

The necessary sensor magnitudes, specifically s and Δv, can be extracted from a radar sensor placed at the front of the car. Particularly, this sensor is defined to have a range of 200 m and a vertical and horizontal aperture of 8º. In CarMaker this sensor can be modelled as an object sensor as follows:

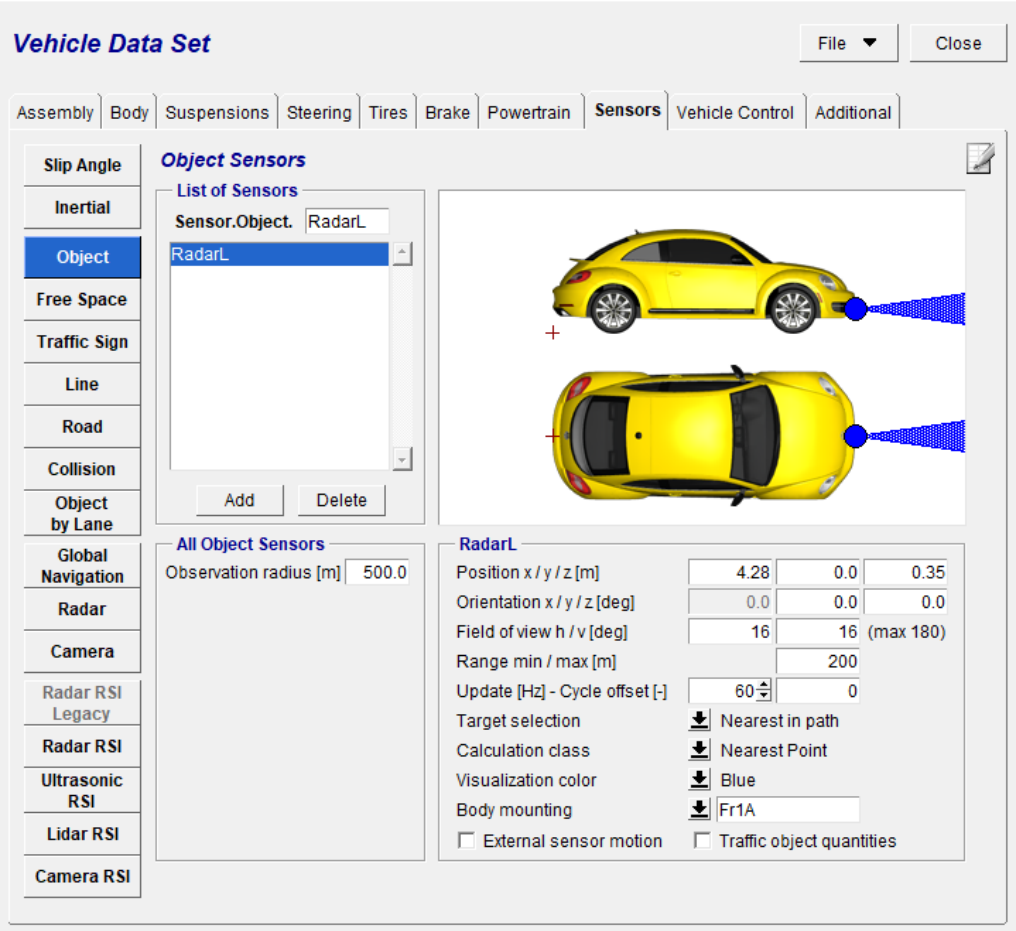

Figure 3-4: CarMaker's object sensor for radar detections

As it can be seen, the target selection is set to ‘Nearest in path’. This means that the sensor will only detect the traffic objects placed in the estimated path calculated by CarMaker. This target selection assumes that the raw data received from the radar sensor has been processed and categorised to not only detect the objects in the estimated path but also only focus on the nearest traffic object.

Also, since the acceleration is overwritten by the acceleration controller in Simulink, in this case the IDM, CarMaker has to allow desired acceleration commands directly from the Simulink model. To do so, the Direct Variable Access (DVA) option has to be selected in the vehicle control tab on the vehicle parameter selection in CarMaker.

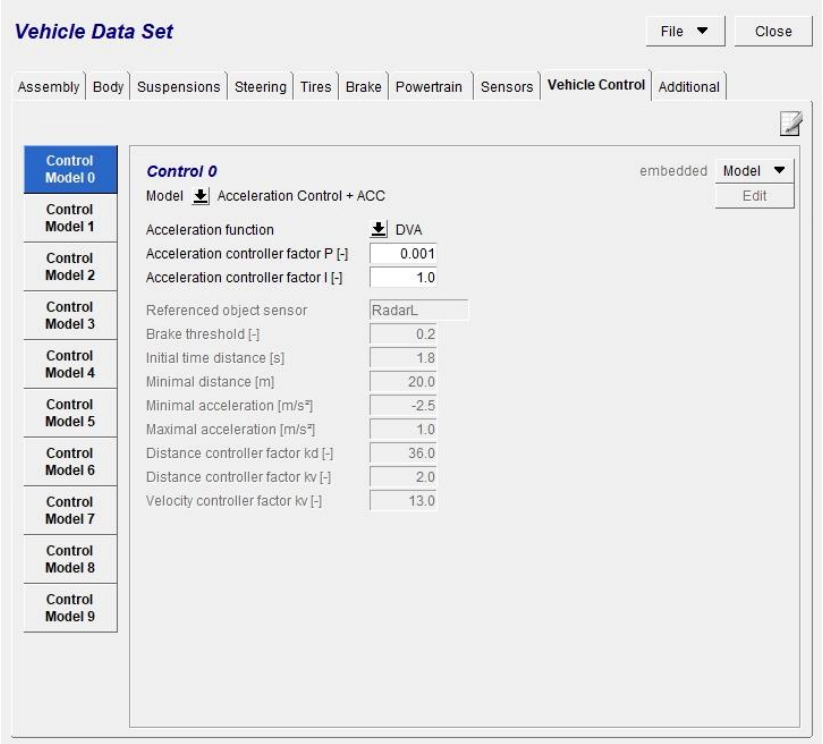

Figure 3-5: Direct Variable Access option for acceleration control

Notice, though, that the formulation proposed in Eq. 2-1 does not work as expected in cases such as in high speed car following scenarios. If the desired speed is not modified when a vehicle is in front of the ego car, the speed ratio starts becoming more relevant as the ego car reaches the desired speed. In turn, to achieve steady state conditions, that is, zero desired acceleration, the distance ratio needs to decrease to consider the action of the speed. To do so, since the desired relative distance, $s^*(v, \Delta v)$, close to steady state conditions mostly depends on the ego vehicle's velocity and, thus, is close to being constant, the only way to decrease the distance ratio is by increasing the measured relative distance. As a consequence, the IDM forces a greater relative distance with the preceding vehicle than the desired one, usually close the road speed limit. To demonstrate this issue, a car following scenario is presented where both the ego and a preceding vehicle start in standstill and slowly approach the maximum road speed, in this case, 130 km/h.

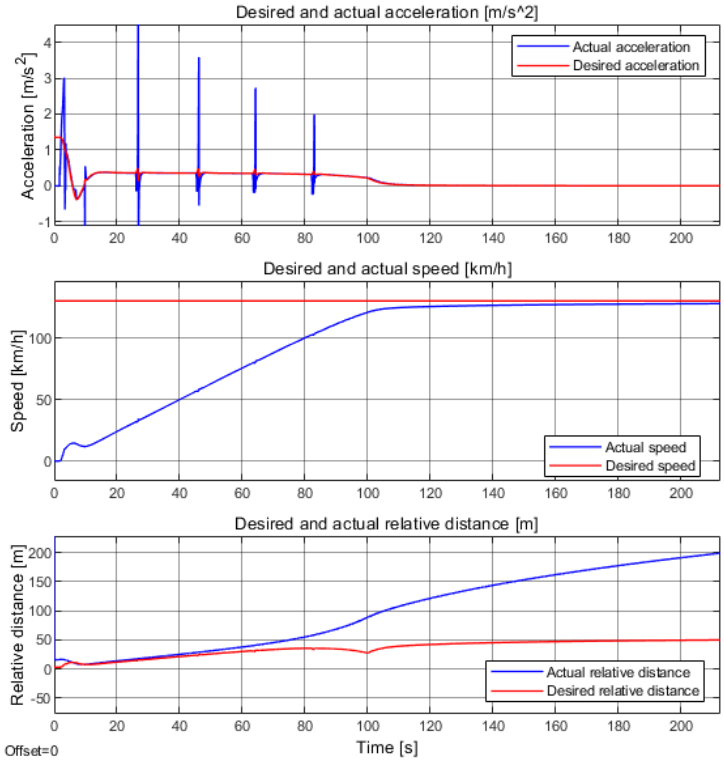

Figure 3-6: Acceleration, speed and relative distance analysis of an accelerating car following scenario with a pure IDM acceleration control

Figure 3-6 clearly demonstrates the behaviour previously explained, where the actual relative distance s needs to increase to reduce the significance of the distance ratio as the speed ratio approaches one. As a clarification, the spikes seen in the actual acceleration plot are a consequence of the gear changes that CarMaker does when the ego vehicle's engine rpm are between a specified rpm range.

To address this issue, the chosen improvement consists in increasing the desired speed whenever there is a traffic object inside the sensor range to reduce the significance of the speed ratio. This improvement, though, has the drawback that, if the preceding traffic object decides to go faster than the original desired speed, usually the maximum speed limit for a specific road, the ego car will follow. To control this behaviour, a substitute for the IDM is used under certain conditions.

The substitute for the acceleration controller consists in a simple P controller that takes as a reference the actual ego car speed and compares it to the road speed limit, so that whenever the car goes close to or beyond the speed limit, the acceleration provided will make it converge to it.

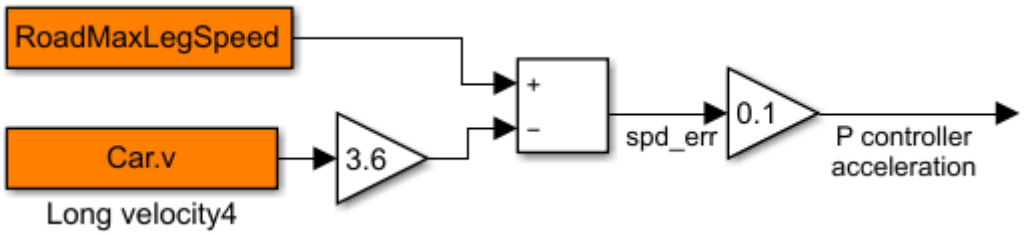

Figure 3-7: Acceleration controller substitute (P controller)

Regarding the conditions that need to be met to switch from the IDM to the P controller, both the ego car's actual velocity and the relative velocity need to be monitored. The only instance where the acceleration manager choses to substitute the desired acceleration by the P controller is if the ego car goes faster than the speed limit and the relative velocity is positive. These conditions describe a scenario where the preceding car is going faster than the limit and the ego car is trying to follow it. If, for instance, the ego car goes beyond the speed limit but the relative velocity is negative, the preceding vehicle is going slower or equal to the speed limit and the IDM needs to surpass the road limit to achieve the desired relative distance, s*(v, Δv). In the latter case, the IDM controller commands the desired acceleration to reach the desired relative distance.

$$\dot{v} = \begin{cases} P\ controller, & if \begin{cases} car\ speed \geq limit \\ AND \\ rel.vel > 0 \end{cases} \\ Intelligent\ Driver\ Model, & else \end{cases} \qquad \text{(Eq 3-2)}$$

Nevertheless, as a safety measure and a feature enabler, whenever the ego car goes over the speed limit and the relative velocity is positive, the minimum value between the IDM and the P controller is chosen. This ensures that if a braking event occurs, the IDM acceleration would be more negative than the P controller's and the desired acceleration would be overwritten by the IDM command. Running the same test case as in the pure IDM controller has the following results:

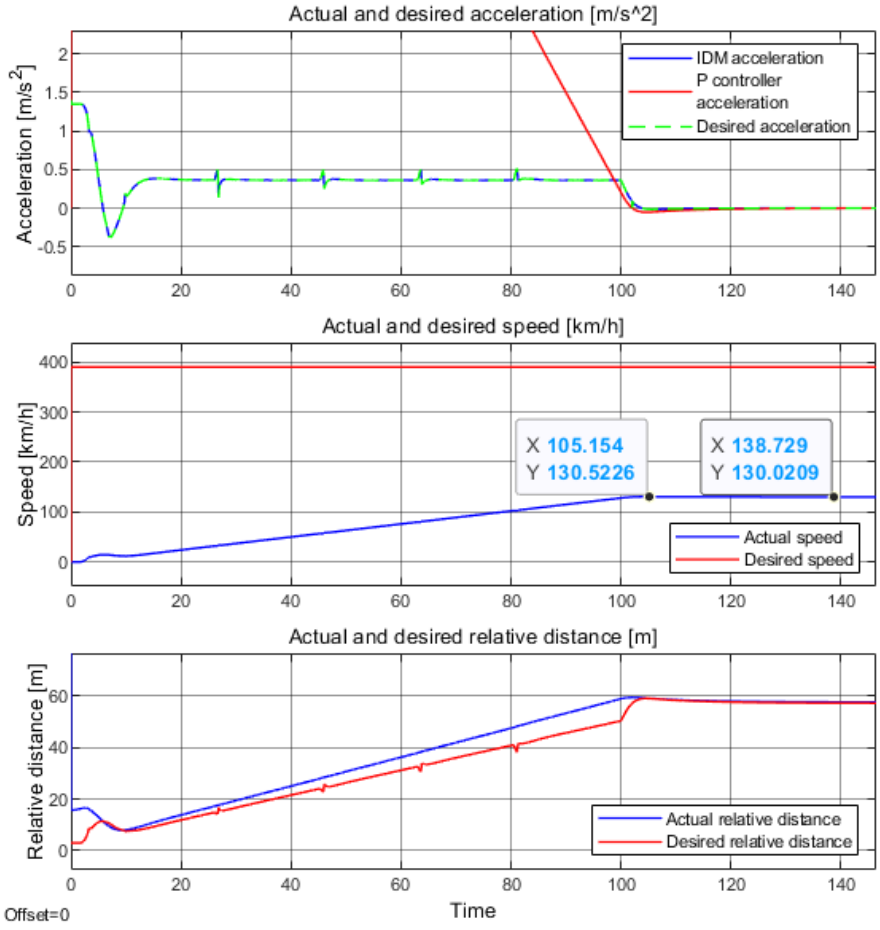

Figure 3-8: Acceleration, speed and relative distance analysis of an accelerating car following scenario with a combined acceleration control

With this improvement, it can clearly be seen that the relative distances proposed by the IDM controller are followed even at high speed scenarios. Notice that the IDM requires speeds higher than the speed limit to achieve those distances. It is in that particular instance when the logic proposed is useful. In this case, since the preceding vehicle stopped accelerating when it reached the speed limit, the relative velocity is zero or negative when the ego car surpasses the speed limit. If the preceding car chooses to develop faster velocities, the relative velocity would be positive and thus the P controller would take command of the acceleration and make the ego car converge to the speed limit.

3.2.2 Lane Keeping Assist

For lane keeping assistance, two controllers where tested while developing the combined ADAS functionality, based on PID controllers. Both controllers take information from a line sensor from CarMaker and extract some type of error, distance or angular, to input

to a PID. The line sensor provides a set of points defining the detected lines and gives relevant information about them, such as their relative position, type, etc. This sensor emulates the capabilities of a camera sensor with some line detecting algorithm incorporated to reliably identify lane bounds Thus, this sensor assumes a proper line detection algorithm that achieves a consistent line detection with minimal time delays. Particularly, the sensor has the following properties:

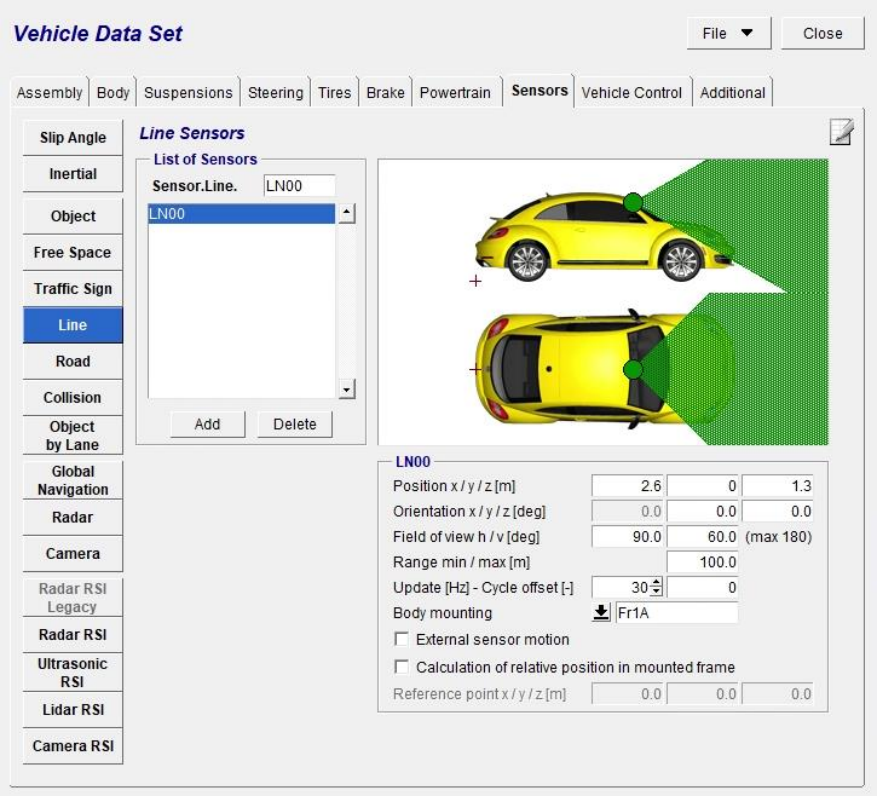

Figure 3-9: Line sensor properties for the LKA controller

As seen in Figure 3-9, now the aperture, vertical and horizontal, are much wider than the radar scope to represent the capabilities of a camera sensor mounted behind the windshield rear-view mirror.

Using this sensor, the first controller implemented during the development process of the project consisted in taking the lateral deviation from the sensor, placed in the vehicle centreline, of two points at the same longitudinal distance of the right and left line detections. Then, the average lateral position deviation was extracted and inputted to the PID controller, in particular, a PD controller. PD controllers are widely used for steering applications because the D coefficient usually manages to attenuate the oscillatory behaviour introduced by curve steering overshoots and velocity dependent

oscillations. Nevertheless, this coefficient introduces noticeable noisy behaviour, especially if the error signal is not smooth or has quantization from signal digitalization. On the contrary, the I coefficient, usually introduced to address steady state errors is hardly ever used, as steady error is non-existent or so small that can be considered negligible. Also, this coefficient introduces oscillation and overshoots which, in this case, are not desirable as overshoots might generate oversteering/understeering due to excess/lack of steering angle commanded. This controller had the following schematic in Simulink:

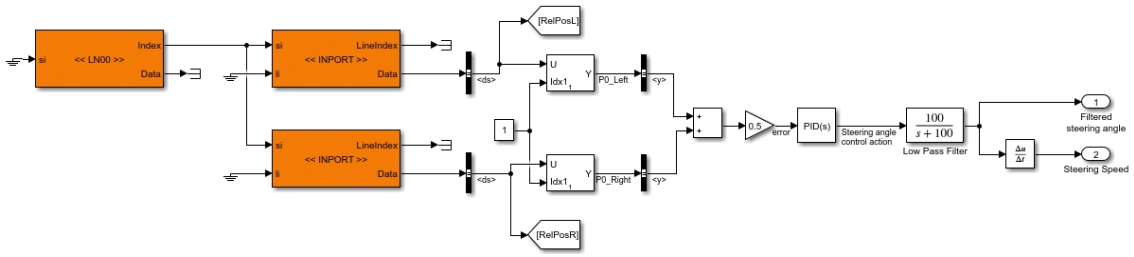

Figure 3-10: Steering angle and velocity control from PD controller

As it can be seen, the points selected from the lines are the first ones detected in the right and left lanes (represented by the constant block with 1 as the index pointer). Since the point detections of the line sensor are equally spaced from each other by 2 meters, the first point in the lines detected are at two meters from the sensor, right in front of the vehicle bumper. Notice, also, that the output of the PID has been passed through a low pass filter to attenuate the noisy behaviour introduced mainly by the D coefficient. This filter reduces the amplitude of all the signal frequencies higher than the natural frequency of the filter, in this case 100 rad/s. By doing so, the comparison between both outputs is the following:

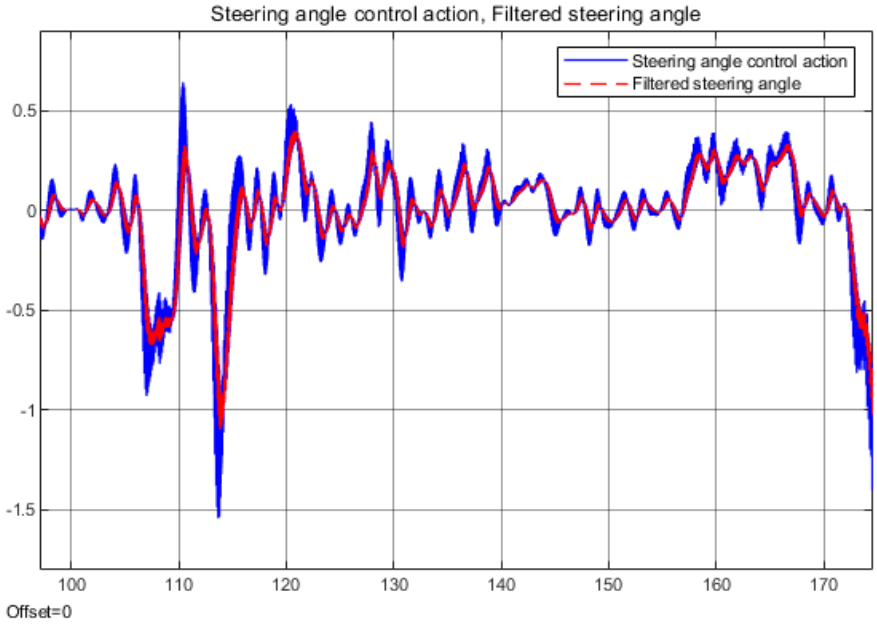

Figure 3-11: Filtered (red) vs unfiltered (blue) steering signals

As it can be seen the noisy signal can benefit a lot from this low pass filter as this reduction in noise means that the steering wheel would jitter less around the desired steering angle and the velocity calculation by deriving the steering angle signal would be less noisy.

Since this controller only considered the lateral deviation, that is, only one degree of freedom of the whole vehicle pose, a more complex controller is implemented considering also the vehicle heading or yaw error. The heading error is calculated by measuring the yaw angle of the car and comparing it to the yaw angle of a point in the path followed. To extract the yaw angle of the path followed two routes can be followed, either calculating the distance increments between two longitudinally adjacent points and calculating the angle with those magnitudes, or fitting a curve through the path followed (lane centreline) and calculate the angle by deriving the fitted curve.

Since the first method depends on the resolution of the points offered by the line sensor and the position variability of these can affect the angle readings, the second method will be implemented. Nevertheless, to use this method, a precise curve needs to be fitted, ensuring that the curve generated is as close as the real lane centre as possible, passing through desired control points. For this task, first, piece-wise Bezier curves were implemented. Despite being beneficial for the ACC development, the curvature readings were not matching the curvature readings native from CarMaker due to a naïve way of implementing these curves, as seen in Figure 3-12.

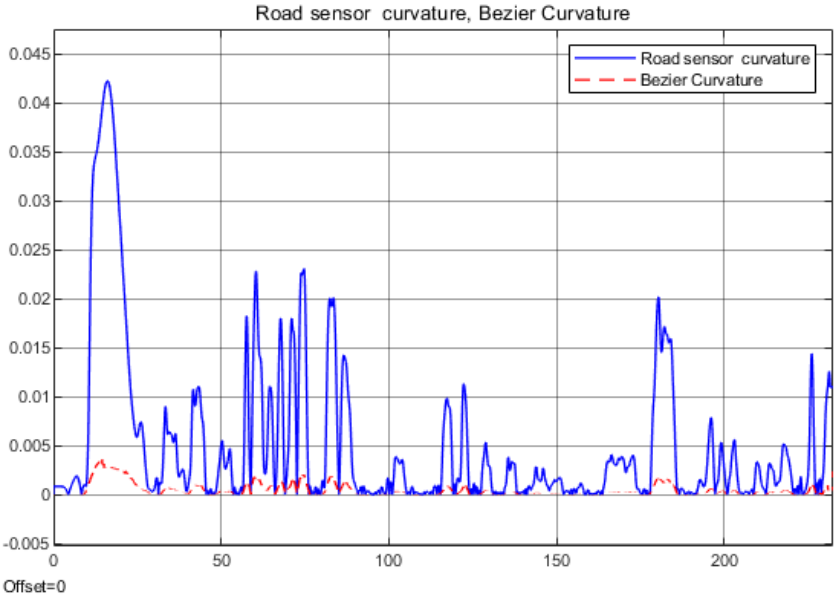

Figure 3-12: Curvature readings from CarMaker's road sensor (blue) and fitted Bezier curves (red)

Notice that the curvature read from a point in the Bezier curves fitted is way lower than the ones presented by the native road sensor from CarMaker. Due to this discrepancy between these magnitudes that should be similar, the fitted curves used for this problem are substituted by Catmull-Rom curves.

Catmull-Rom curves

Catmull-Rom curves take four control points and define a curve that passes through the middle two points. This feature is really helpful due to the fact that, by only by defining control points that go along the lane centreline, a precise curve can be fitted through them. The curves, C(t), are defined by the following equations:

$$C(t) = \frac{t_2 - t}{t_2 - t_1} \cdot B_1(t) + \frac{t - t_1}{t_2 - t_1} \cdot B_2(t) \quad \text{(Eq 3-3)}$$

$$B_1(t) = \frac{t_2 - t}{t_2 - t_0} \cdot A_1(t) + \frac{t - t_0}{t_2 - t_0} \cdot A_2(t) \quad \text{(Eq 3-4)}$$

$$B_2(t) = \frac{t_3 - t}{t_3 - t_1} \cdot A_2(t) + \frac{t - t_1}{t_3 - t_1} \cdot A_3(t) \quad \text{(Eq 3-5)}$$

$$A_1(t) = \frac{t_1 - t}{t_1 - t_0} \cdot p_0 + \frac{t - t_0}{t_1 - t_0} \cdot p_1 \quad \text{(Eq 3-6)}$$

$$A_2(t) = \frac{t_2 - t}{t_2 - t_1} \cdot p_1 + \frac{t - t_1}{t_2 - t_1} \cdot p_2 \quad \text{(Eq 3-7)}$$

$$A_3(t) = \frac{t_3 - t}{t_3 - t_2} \cdot p_2 + \frac{t - t_2}{t_3 - t_2} \cdot p_3 \quad \text{(Eq 3-8)}$$

$$t_i = t_{i-1} + \sqrt{(x_i - x_{i-1})^\alpha + (y_i - y_{i-1})^\alpha} \quad \text{(Eq 3-9)}$$

Where $\alpha \in [0, 1]$, i = 1, 2, 3, and t_0 is 0. Note that the curve at t_1 has a value of p_1 and at t_2 a value of p_2, as desired. Depending on the value of α, the curve can be categorised into uniform if α = 0, centripetal if α = 0.5, and chordal if α = 1. In some way, α deals with the position when the curvature starts being more significant around the control points to reach the next point. In this case, α = 0.5 is chosen because on top of being the value that follows the control points the tightest, it also has the property that it will not form loops or self-intersect, avoiding undesired fittings.

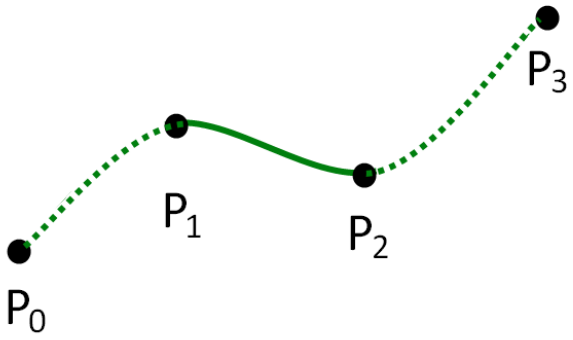

Figure 3-13: Catmull-Rom curves defined between the second and third control points [7]

By concatenating multiple Catmull-Rom segments, a long curve that passes multiple control points can be defined. For the application, two Catmull-Rom curves are fitted onto the path centreline depending on their application and maximum distance requirement.

For LKA, since the point where the lateral deviation and yaw angle are measured does not need to be too far from the ego vehicle, just four control points are defined. By calculating the average lateral position of the right and left lane points at their respective longitudinal distances, the desired control points are placed at the lane centreline. For this application, these points are placed, in order from closer to farthest, at the sensor's origin, at two meters from it, at 22m and, at 24 meters. Notice how the first and the second points are at the same distance as the third and the fourth and a much smaller

distance than from the second to the third. This configuration is taking advantage of Catmull-Rom only passing through the two middle points of a set of four points.

When combining the ACC and LKA functionalities, six control points need to be used for a similar application, measuring the curvature of the path at a much farther distance. The first point is placed at the line sensor's origin, the second at 10 meters from it, the third, fourth and fifth are equally spaced between each other 20 m apart, and the final point is 10 meters from the last one.

By using six points, three Catmull-Rom curves are concatenated using from the first to the fourth point for the first curve, from the second to the fifth point for the second, and from the third to the sixth point for the third curve. Since these curves are only defined between the middle two points, there is no overlap between the three curves and generate a combined curve that passes through the second up to the fifth control point.

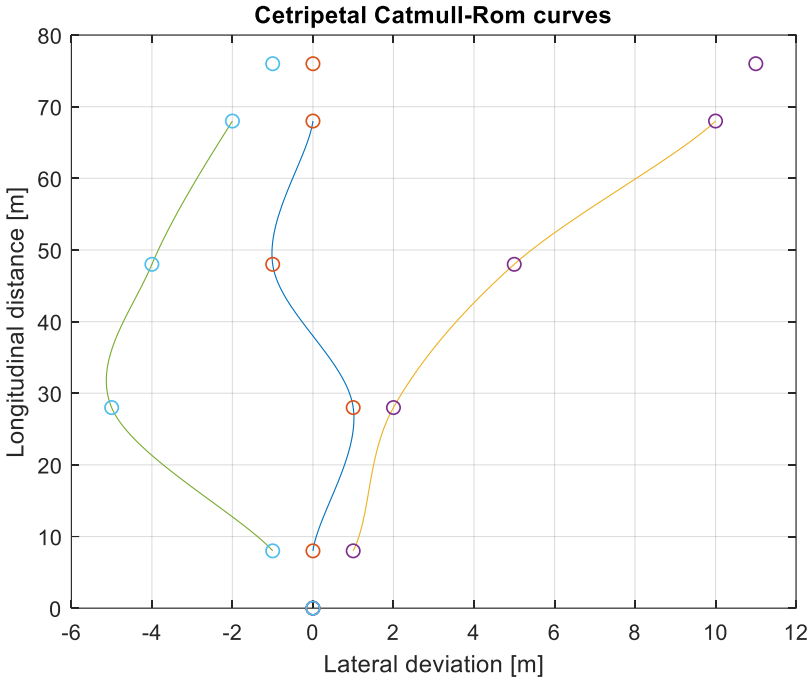

Figure 3-14: Examples of piecewise centripetal Catmull-Rom curves

As it can be seen, for all curves the origin is placed at (0, 0) where the position of the origin of the line sensor is, and the last control point is placed so that it follows the lane centreline detected. It is important to note that these curves are C^1 and, therefore, the curvature is not continuous along the control points which can be problematic if the value differs drastically between two consecutive curves which is not the usual thing to happen.

Now, if the curvature native to CarMaker's road sensor and the one extracted from this strategy is compared at a similar longitudinal distance, it can be seen that the result matches noticeably better than Bezier curves, hence, providing a much better fitted curve.

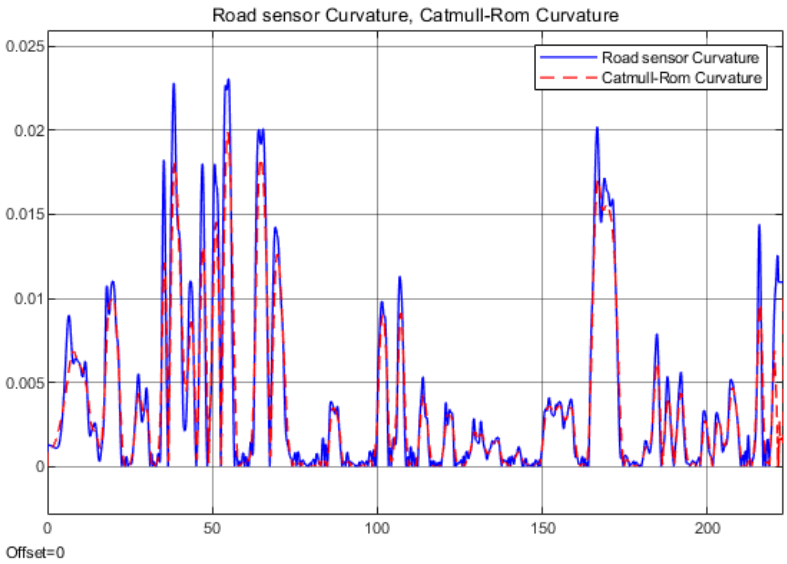

Figure 3-15: Curvature readings from CarMaker's road sensor (blue) and fitted Catmull-Rom curves (red)

In all instances where curvature is measured from a fitted path, it is extracted from using the parametric curvature formula as follows:

$$k = \left| \frac{x' \cdot y'' - y' \cdot x''}{[(x')^2 + (y')^2]^{\frac{3}{2}}} \right| \qquad \text{(Eq 3-10)}$$

Where k is the curvature of a point in the fitted curve. x' and x'' are the first and second derivative of the longitudinal curve expression, x, respectively, while y' and y'' are the same magnitudes from the lateral curve expressions, y, at that point

Once these curves are defined using the control points extracted from the lane marking readings, the yaw angle can be calculated by deriving the curve equations with respect to t and using the x and y variations. The expression of the angle can be extracted from the following equations.

$$A_1' = \left(\frac{1}{t_1 - t_0}\right) \cdot (p_1 - p_0) \qquad \text{(Eq 3-11)}$$

$$A_2' = \left(\frac{1}{t_2 - t_1}\right) \cdot (p_2 - p_1) \quad \text{(Eq 3-12)}$$

$$A_3' = \left(\frac{1}{t_3 - t_2}\right) \cdot (p_3 - p_2) \quad \text{(Eq 3-13)}$$

$$B_1'(t) = \left(\frac{1}{t_2 - t_0}\right) \cdot (A_2(t) - A_1(t) + (t_2 - t) \cdot A_1' + (t - t_0) \cdot A_2') \quad \text{(Eq 3-14)}$$

$$B_2'(t) = \left(\frac{1}{t_3 - t_1}\right) \cdot (A_3(t) - A_2(t) + (t_3 - t) \cdot A_2' + (t - t_1) \cdot A_3') \quad \text{(Eq 3-15)}$$

$$C'(t) = \left(\frac{1}{t_2 - t_1}\right) \cdot (B_2(t) - B_1(t) + (t_2 - t) \cdot B_1'(t) + (t - t_1) \cdot B_2'(t)) \quad \text{(Eq 3-16)}$$

These expressions all have two components, one for the longitudinal, $C_x(t)$, and for the lateral, $C_y(y)$, axes. Using these components from Eq. 3-16 at a certain point at t, the relative yaw angle of the path to the car can be extracted as:

$$\psi_{path}\big|_{car} = \arctan\left(\frac{C_y'}{C_x'}\right) \quad \text{(Eq 3-17)}$$

Since CarMaker provides the yaw angle of the car in a fixed frame of reference, the road yaw angle with respect to the car needs to be converted to that frame of reference. To do so, the angle of the car needs to be added to the path's yaw angle resulting in the following expression:

$$\psi_{path}\big|_{abs} = \psi_{path}\big|_{car} + \psi_{car}|_{abs} \quad \text{(Eq 3-18)}$$

Nevertheless, since the heading error used for the steering controller is calculated by subtracting the car's yaw angle to the curve's one, the error can be directly extracted by calculating the yaw angle of the path relative to the car e_ψ.

$$e_\psi = \psi_{path}\big|_{abs} - \psi_{car}|_{abs} = \psi_{path}\big|_{car} = \arctan\left(\frac{C_y'}{C_x'}\right) \quad \text{(Eq 3-19)}$$

Once this error is obtained, the lateral deviation error can be extracted by comparing the lateral position of the same point in the fitted curve with the position of the sensor. Also, due to this calculation being done in the car's frame of reference, the lateral position of the fitted point is directly the lateral error e_δ.

$$e_\delta = \delta y_{path}\big|_{car} + py_{car}|_{abs} - py_{car}|_{abs} = \delta y_{path}\big|_{car} \quad \text{(Eq 3-20)}$$

Where $\delta y_{path}\big|_{car}$ is the lateral deviation relative to the car and $py_{car}|_{abs}$ is the absolute car position in the y axis. $\delta y_{path}\big|_{car}$can be extracted directly from the lateral coordinate from Eq. 3-3:

$$\delta y_{path}\big|_{car} = C_y \quad \text{(Eq 3-21)}$$

With these two errors, the two degrees of freedom are contemplated and, thus, a PID can be put in series with each error to generate a control action for the steering wheel angle. In this case, just using a proportional gain for each error proves to be the best outcome possible, as the D coefficient adds too much unnecessary noise and the I introduces oscillations and overshoots in the angle. Once the two control actions are calculated with its proportional gains, there needs to be a way to combine these two into one to make both control the steering angle. For this, several algorithms such as the gain scheduling method presented in [5] have been tested, but the most successful one ended up being just adding both errors and carefully tuning the proportional coefficients. The gain scheduling method in [5] introduced external effects to the PID controller that made tuning the controller unpredictable and, overall, difficult. For this, it was deprecated and considered just the addition of both control actions.

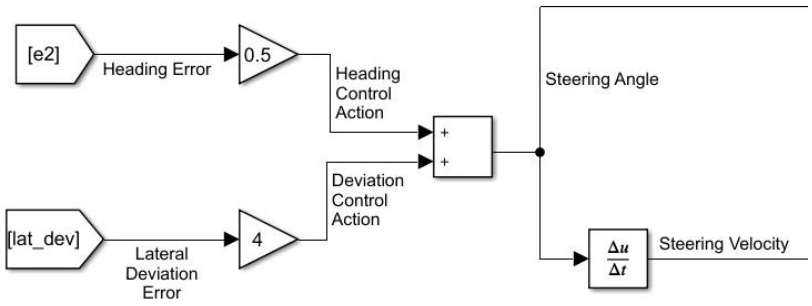

Figure 3-16: Lateral controller CarMaker implementation

Note that the output of the controller is not passed through the low pass filter implemented in the previous controller developed due to the lack of a derivative term adding noise to the steering angle signal. Also, note that the steering speed is controlled by deriving the steering angle signal. In order to overwrite the steering controller that comes by default in CarMaker, the steering model needs to be turned off.

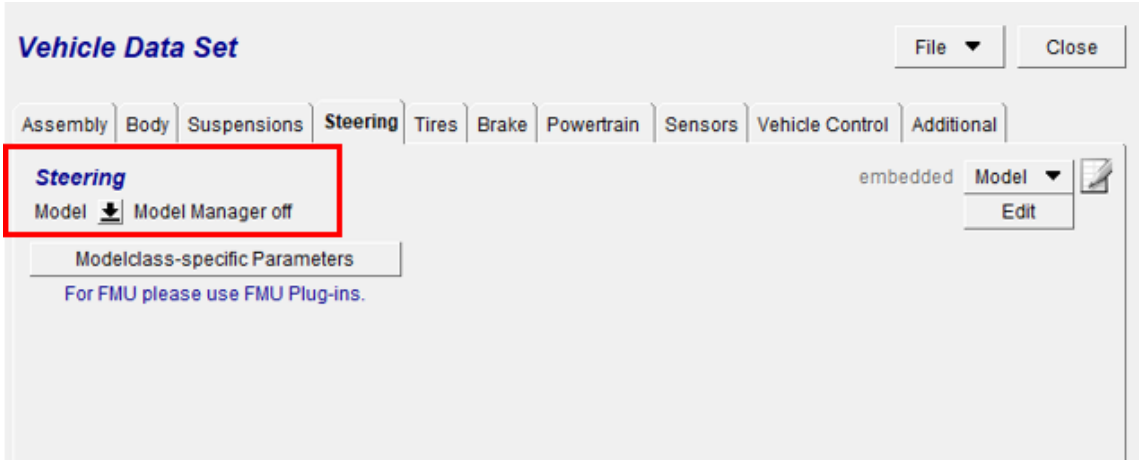

Figure 3-17: CarMaker vehicle changes for steering control

Also, the user defined steering block from the 'UserSteer.mdl' from the provided examples in CarMaker has to be placed inside the steering block in Simulink, the same as in the example model.

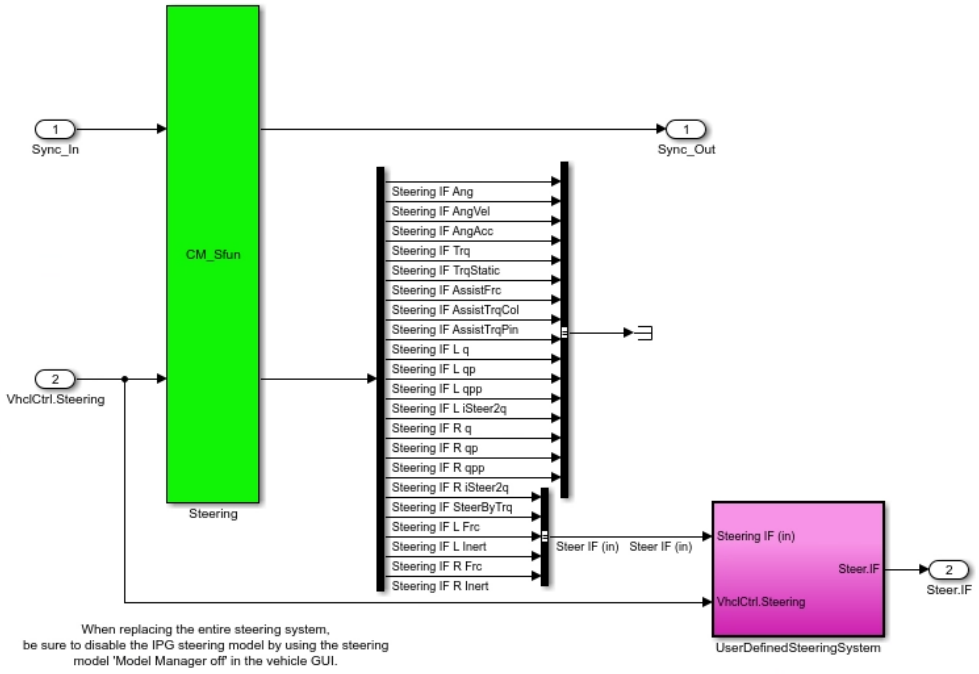

Figure 3-18: Simulink changes for steering control

3.2.3 Combined ACC and LKA functionality

Once both controllers were developed and tested separately, an implementation that merged both controllers was devised. While testing the ACC functionality on its own, the steering was completely controlled by Carmaker and, for the LKA functionality, the

acceleration was controlled by the simulating software. This is why, when merging both controllers into the same application, the aid Carmaker provided completely vanished. Now, the acceleration is fully commanded by the ACC and the steering action by the LKA. In turn, this condition posed an interesting challenge of how to modify the velocity and which variables to consider to appropriately approach a curve as safe as possible.

After research was conducted, several combined longitudinal and lateral controllers seemed to have similar behaviours and dependencies. The majority of them considered the curvature of the path followed to modify the ego vehicle's speed or set the maximum speed for a specific curve. Generally, the curviest a path is, the slower the vehicle has to be and vice versa. The curvature of a path, though, is a property of a point within a curve and, thus, selecting the position of this point relative to the vehicle is also an interesting problem to face.

In this case, [5] has been analysed and tested, yielding the best results. This approach sets the vehicle's maximum velocity as the inverse of the curvature of a moving point in the curve. The maximum velocity can be derived from the lateral acceleration equation, Eq. 2-3, as in [5]:

$$U_{max} = \sqrt{\frac{a_{max}^{lat} \cdot \mu_y}{|\rho|}} \qquad \text{(Eq 3-22)}$$

Where U_{max} is the maximum velocity appropriate for the curvature, a_{max}^{lat} is the maximum lateral acceleration possible of the vehicle, μ_y is the road's lateral coefficient of friction and, finally, ρ is the curvature of the path at a certain point in it.

Further analysis on this formula shows that the higher the curvature is, the lowest the maximum speed required, as desired. Nevertheless, when the curvature approaches zero, this maximum velocity becomes infinite and, thus, the minimum velocity between this and the road's speed limit needs to be chosen for the desired speed.

The position of the point where the curvature is measured is set as speed dependent, ensuring that when the vehicle goes at higher speeds the curvature is measured at longer distances, providing more predictive behaviour. For this feature to be implemented, the following expression is used [5]:

$$s_p = \frac{U^2}{2 \cdot a_{max}^{brake} \cdot \mu_x} \qquad \text{(Eq 3-23)}$$

Where s_p is the distance of the curvature measuring point with respect to the curve's origin, U is the car's actual velocity, a_{max}^{brake} is the maximum braking deceleration possible by the vehicle, and μ_x is the longitudinal friction coefficient of the road. Note that this equation depends on the square of the velocity making small velocities increase the curvature measuring point distance less that higher velocities do.

Since a curve is fitted into the trajectory followed, by deriving the curve's equation twice with respect to t, the curvature of the path ahead can be extracted as follows:

$$C(t) = \frac{t_2 - t}{t_2 - t_1} \cdot B_1(t) + \frac{t - t_1}{t_2 - t_1} \cdot B_2(t) \quad \text{(Eq 3-3)}$$

$$C'(t) = \left(\frac{1}{t_2 - t_1}\right) \cdot \left(B_2(t) - B_1(t) + (t_2 - t) \cdot B_1'(t) + (t - t_1) \cdot B_2'(t)\right) \quad \text{(Eq 3-15)}$$

$$C''(t) = \left(\frac{1}{t_2 - t_1}\right) \cdot \left(2 \cdot (B_2'(t) - B_1'(t)) + (t_2 - t) \cdot B_1''(t) + (t - t_1) \cdot B_2''(t)\right) \quad \text{(Eq 3-24)}$$

$$B_1''(t) = \left(\frac{2}{t_2 - t_0}\right) \cdot (A_2'(t) - A_1'(t)) \quad \text{(Eq 3-25)}$$

$$B_2''(t) = \left(\frac{2}{t_3 - t_1}\right) \cdot (A_3'(t) - A_2'(t)) \quad \text{(Eq 3-26)}$$

By evaluating the derivatives at a distance s_p calculated using Eq. 3-23 and using the parametric curvature formula in Eq. 3-10, the curvature of the path can be extracted. To convert from the point's distance, s_p, to the Catmull-Rom curves' t variable, s_p is linearly interpolated between t_1 and t_2 of the curve where s_p lays, which corresponds to p_1 and p_2 of the segment, respectively.

$$t = \frac{t_2 - t_1}{p_2 - p_1} \cdot (s_p - p_1) + t_1 \quad \text{(Eq 3-27)}$$

In CarMaker, the implementation of the speed controller and the variable distance of the curvature measuring point look as follows:

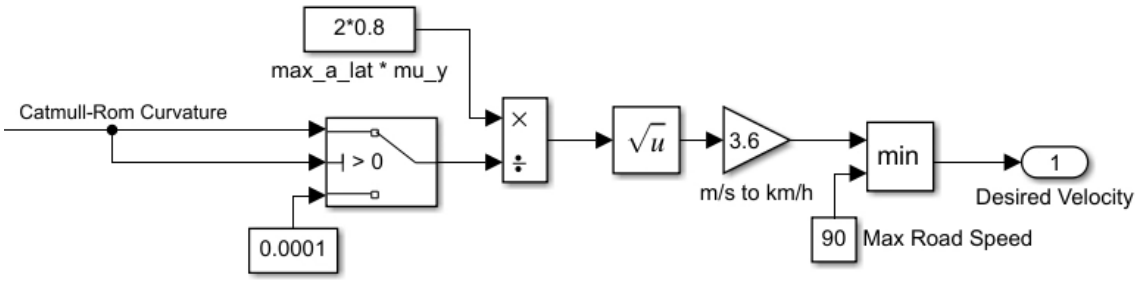

Figure 3-19: Curvature dependent speed controller

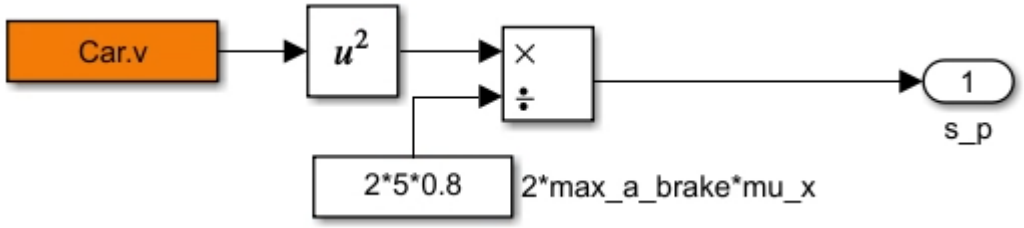

Figure 3-20: Curvature measuring point variable distance controller

In this case, since the output of this method to approach curves is a speed limit and the IDM has a desired speed term, it can be directly connected there. Therefore, the ACC controller will not only handle distance management with the preceding vehicle but also will be able to regulate the speed to approach curves appropriately.

Note that the parameters used for the controllers are vehicle dependent in the case of the maximum lateral and braking accelerations; and road dependent in the case of the lateral and longitudinal friction coefficients. Also notice that, in this case, the longitudinal and lateral friction coefficients have been assumed to be equal and constant, both being quite imprecise, as longitudinal and lateral friction coefficients depend differently on tyre longitudinal slip. Finally, since the car being simulated is generated by Simulink, the lateral acceleration has been extracted from [8] to be within the range of primary handling (between 0 and 3 m/s^2) and the maximum braking acceleration has been calculated from analysing the maximum acceleration possible for a FWD vehicle in an horizontal road without wind.

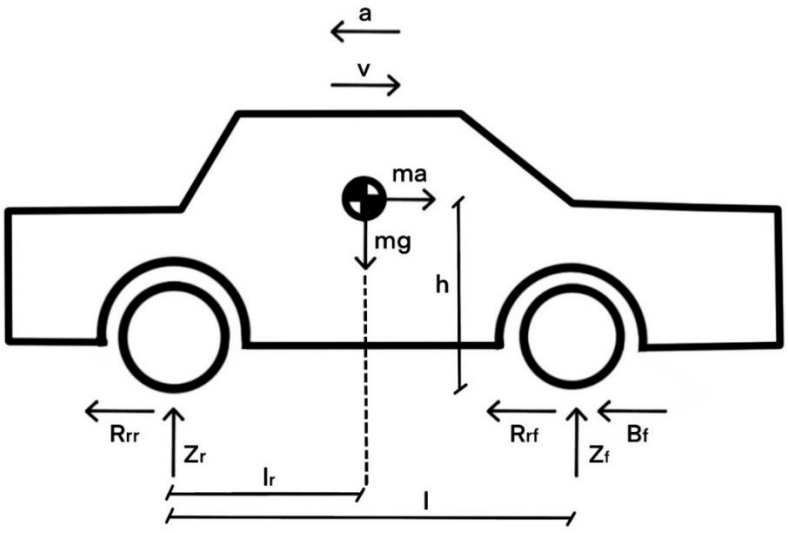

Figure 3-21: Car analysis for maximum braking parameter

Note that, for this case, since velocity is low in full-stop braking scenarios, air resistance is considered negligible and rolling resistance constant. Taking into consideration these assumptions, the maximum braking deceleration can be expressed as:

$$B_f + R_{rr} + R_{rf} = m{\cdot}a \quad \text{(Eq 3-28)}$$

$$\mu_x{\cdot}Z_f + f{\cdot}\,(Z_f + Z_r) = \mu_x{\cdot}Z_f + f{\cdot}\,m \cdot g = m{\cdot}a \quad \text{(Eq 3-29)}$$

$$Z_f{\cdot}\,l - m \cdot g \cdot l_r - m \cdot a \cdot h = 0 \quad \text{(Eq 3-30)}$$

$$Z_f = \frac{m \cdot g \cdot l_r + m \cdot a \cdot h}{l} \quad \text{(Eq 3-31)}$$

By substituting this last expression into Eq. 3.29, the maximum braking deceleration can be found as:

$$a = g \cdot \frac{\mu_x \cdot l_r + f \cdot l}{l - \mu_x \cdot h} \quad \text{(Eq 3-32)}$$

With this expression, and using the vehicle's parameters from CarMaker, the maximum braking acceleration ends up being:

$$a = g \cdot \frac{\mu_x \cdot l_r + f \cdot l}{l - \mu_x \cdot h} = 9.81 \cdot \frac{0.8 \cdot 2.406 + 0.01 \cdot 4.28}{4.28 - 0.8 \cdot 0.566} = 5.04 \frac{m}{s^2} \quad \text{(Eq 3-33)}$$

With these parameters set, the baseline for the combined functionality with adaptive cruise control and lane keeping assist is completed. Nevertheless, to add some safety improvements considering the sensor configuration (one radar sensor and one line-detecting sensor), an additional safety strategy has been implemented.

3.2.4 Crash prevention when a traffic object gets out the sensor range (Variable horizon)

Since the ACC is dependent on whether or not an object is detected, that is, if it is inside the sensor scope or not, it is interesting to study what should the algorithm do when a traffic object suddenly goes out of the sensor range, for instance in curve scenarios. This feature is devised to prevent the ego vehicle from accelerating to the desired speed whenever the detected object goes out of range.

To take into consideration these particular cases, the intersection between the sensor cone and the path fitted is monitored so that whenever a vehicle goes out of the sensor, the vehicle expects a car to re-enter the sensor cone through that point. To implement this behaviour, the distance between the sensor origin and the intersection is fed into the Intelligent Driver Model as the actual relative distance. Thus, the ego vehicle adapts the acceleration to expect a vehicle through that point with zero relative velocity.

In other words, whenever the preceding vehicle goes out of the lateral sensor scope, this feature considers the intersection between the path and the sensor as a vehicle without relative motion to the ego car. This is why, this strategy can help specially in tight curves scenarios where the preceding vehicle suddenly decides stops or an exogenous traffic element appears in the middle of the lane, as the algorithm can expect it and slow down smoothly and safe.

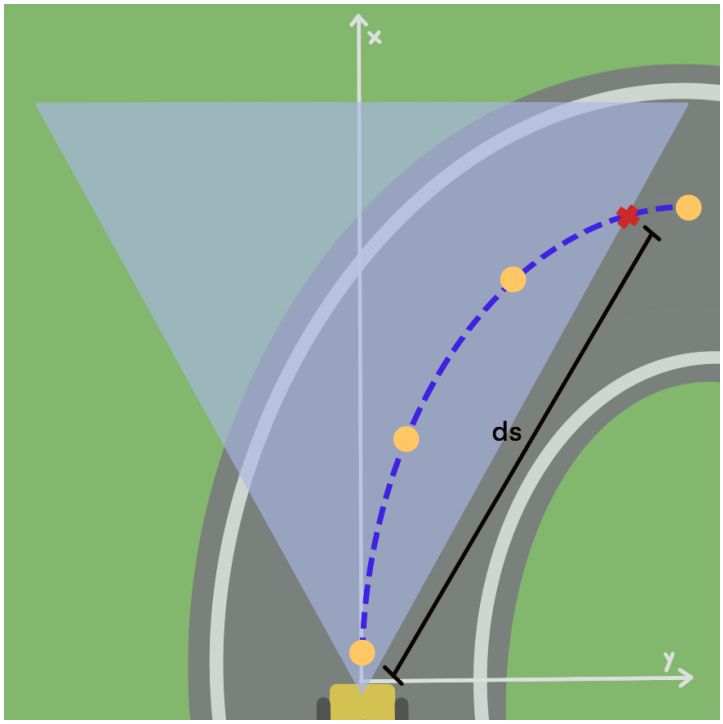

Figure 3-22: Crash prevention algorithm when a traffic object exceeds the sensor range diagram

However, due to this feature being dependent on the intersection of the sensor cone and the path fitted, the aperture of the sensor can impact significantly the performance of this algorithm. In particular, tighter apertures mean that the sensor bounds are more vertical and, hence, lateral variations of the intersection position imply larger longitudinal variations. This, in turn, causes the distance to the intersection point to quickly decrease to low values, which can cause sudden braking, even tyre slip.

To combat this, it would be recommendable to use this feature with a superposition of object detecting sensors with wider aperture angles, for instance, a radar plus a camera or radar plus LIDAR. In addition, this implementation could be triggered only whenever a traffic object goes from being detected to not being detected. Though, the latter option, would not be desirable as a traffic object can suddenly appear in a tight turn even if it has not been detected previously, for instance a fallen rock.

4 Results

To assess the validity of the controller designed, several test cases with varying traffic and road profiles are run to check whether or not the combined functionality is able to withstand the typical cases and real life general scenarios.

First, the typical test cases for adaptive cruise control (approach, separate, cut in/out, etc.) will be evaluated and analysed isolated from the lane keeping assist effect, that is, in straight roads.

Second, to test the lane keeping assist algorithm, the lateral deviation from the lane centreline of the vehicle's centre of gravity will be monitored in different road profiles. Also, the yaw angle of the vehicle and the road at the vehicle's centre of gravity will be compared. If the controller works properly, the results of comparing these magnitudes should always oscillate around zero, denoting that the car is correctly oriented at the middle of the lane. Also, since there has been the necessity to add acceleration control for curve approaching, these test runs will also be useful to assess the acceleration changes for different curve profiles.

Finally, to evaluate the result of combining both implementations, a general test case will be run. This general test case will include stochastic traffic and challenging road scenarios that will ensure a proper assessment of both functionalities combined.

This paper, though, will not go through neither the full validation process nor the evaluation of edge cases (loss of sensor signal, sensor/controller lags, etc.) as it is not the purpose of it.

4.1 Adaptive cruise control results

To evaluate the ACC completely isolated from the LKA, test runs of the common cases will be run in the simplest scenario possible for the steering controller, a straight road. The test cases evaluated are a subset from the ones presented in [2], as the combined functionality does not have incorporated the capability of lane changes. Thus, the test cases considered will be the ones shown in Figure 4-1 except the lane change scenario.

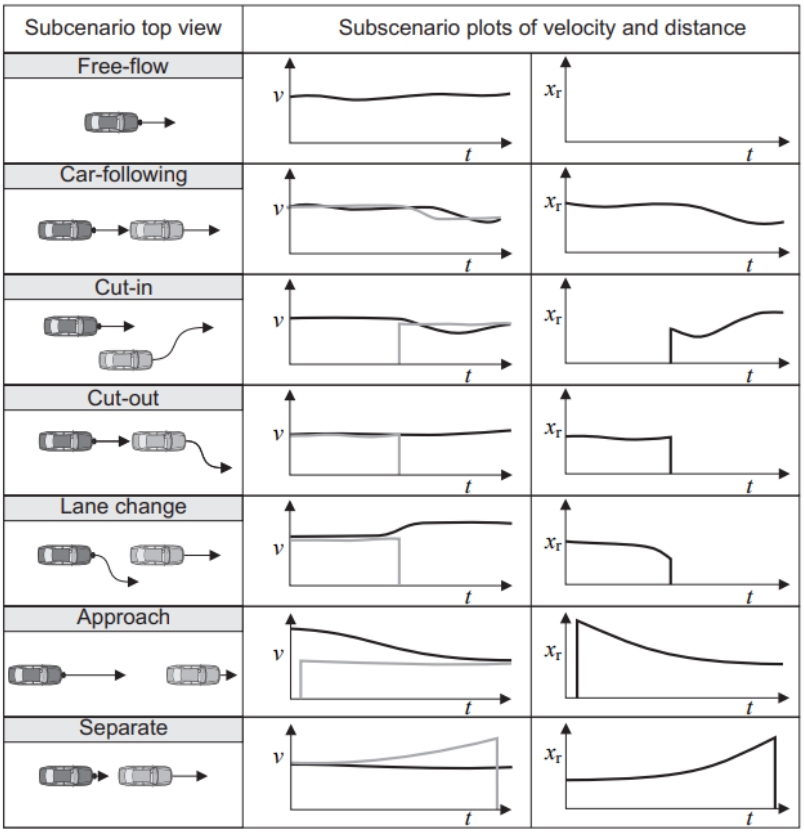

Figure 4-1: Typical test cases for an adaptive cruise controller [2]

4.1.1 Free flow

To make the test more interesting, the vehicle will start from a complete standstill and will accelerate up until the desired speed. In this case, since the controller needs to handle properly up to highway speeds, the desired speed will be set to the maximum legal highway speed in Germany, of 130 km/h.

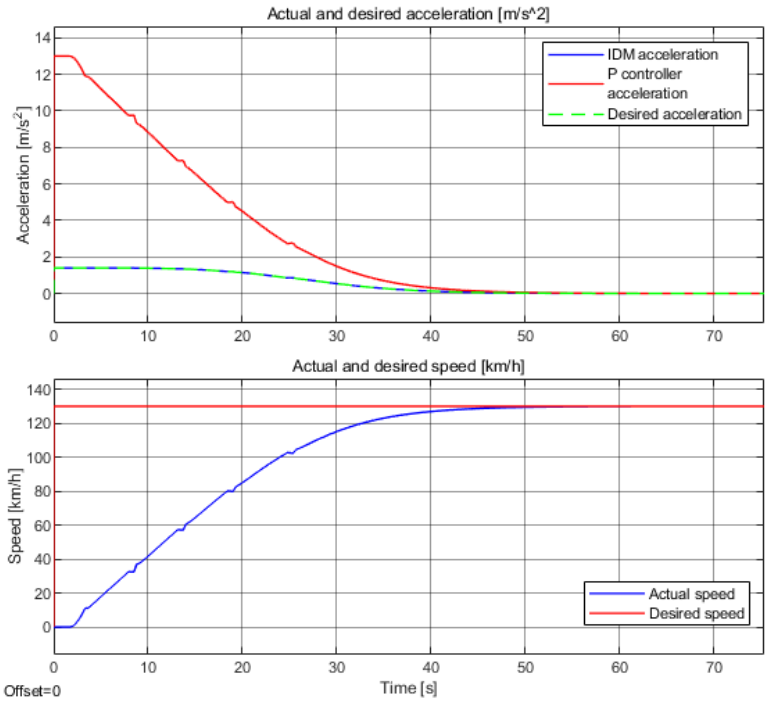

Figure 4-2: Free flow test case starting from standstill

Notice how the acceleration is fully commanded by the IDM controller at all time, since the acceleration switching strategy needs a preceding vehicle to activate it. The absence of a preceding vehicle can be inferred from the plots presented in the results section by analysing the desired speed or the actual relative distance. Due to the IDM implementation, the desired speed is multiplied by three when there is a preceding object in the radar scope. Similarly, the actual relative distance is kept at a really high value when there is not a preceding vehicle in sight so that the distance ratio of the IDM controller is negligible. When a vehicle enters the radar's scope, the actual relative distance is set to the value generated by the sensor readings. Since neither of these instances has occurred in the plots from Figure 4-2, it can be deduced that there is no preceding vehicle.

Also, notice that the actual speed plot has some small variations at around 3s, 8s, 14s, 19s, and 25s. This is an effect generated by the native gear changes CarMaker does when entering a certain rpm range. Notice that the actual acceleration is not represented in the acceleration plot to avoid signal cluttering. If it were, there would be acceleration spikes at those time stamps, representing the gear changes, see Figure 3-6.

4.1.2 Approach, car following and separate

To keep this project more succinct, these three test cases will be evaluated and analysed as a pack. The ego car will be in free flow at the speed limit of 130 km/h while the preceding car will be at a much lower speed, in this case 80 km/h. When the preceding vehicle enters the radar sensor scope, the approach scenario will be evaluated. Then, several seconds of car following will be shown to demonstrate this capability at constant speed. Finally, the preceding car will develop the speed limit and both vehicles should reach it keeping the desired relative distance between both.

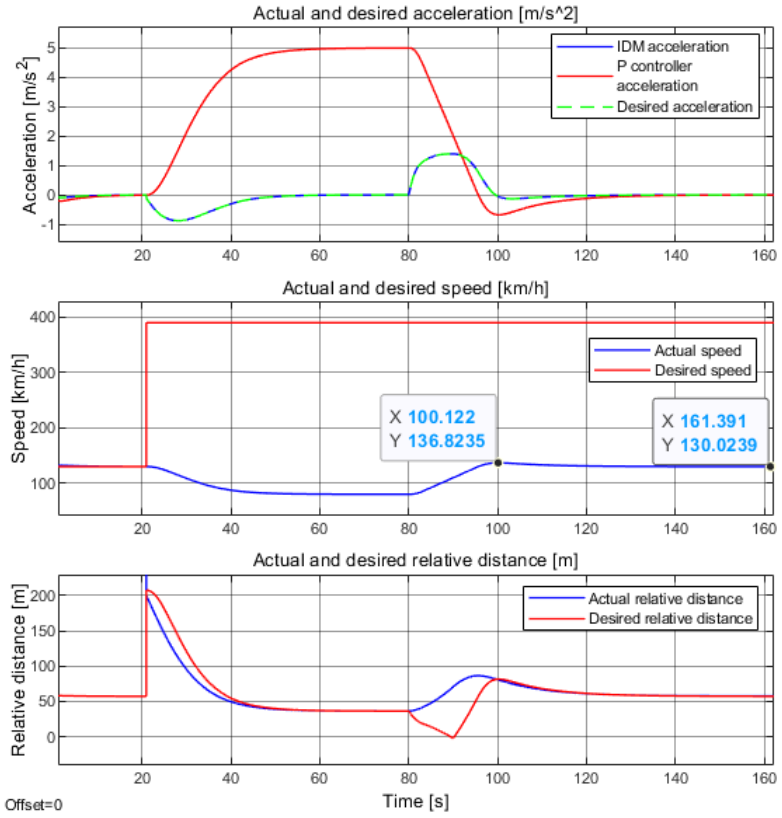

Figure 4-3: Approach, car following and separate combined scenario

These plots represent free flow up to approximately 20s and, thus, no preceding car is in front of the ego vehicle. Notice how up until that timestamp, the desired speed is at 130 km/h and the actual relative distance is not present as it is too large. Then, the preceding vehicle enters the sensor scope at 200m, generating a negative acceleration, as it is going at 80 km/h, 50 km/h slower than the ego vehicle at free flow. Then, up until 80s, the car following scenario is shown where the ego vehicle asymptotically reaches that speed, keeping the desired relative distance tracked. Finally, the preceding vehicle accelerates until the speed limit and stays there. Since it never goes beyond the speed

limit, the relative velocity when the ego car goes beyond it is always negative or zero. Taking advantage of this situation, notice the controller never switches to the P controller to return to the speed limit even if the speed limit is surpassed. This enables the IDM controller to safely reach the desired relative distance proposed.

4.1.3 Car following scenario with the preceding car surpassing the speed limit

This test case is set to demonstrate the capabilities of the implementation to return to the speed limit when the preceding car decides to go beyond the speed limit in a car following scenario. This test consists in both the ego and the preceding car beginning at standstill and the preceding vehicle slowly accelerating beyond the speed limit, up until 170 km/h. This test yields the following results:

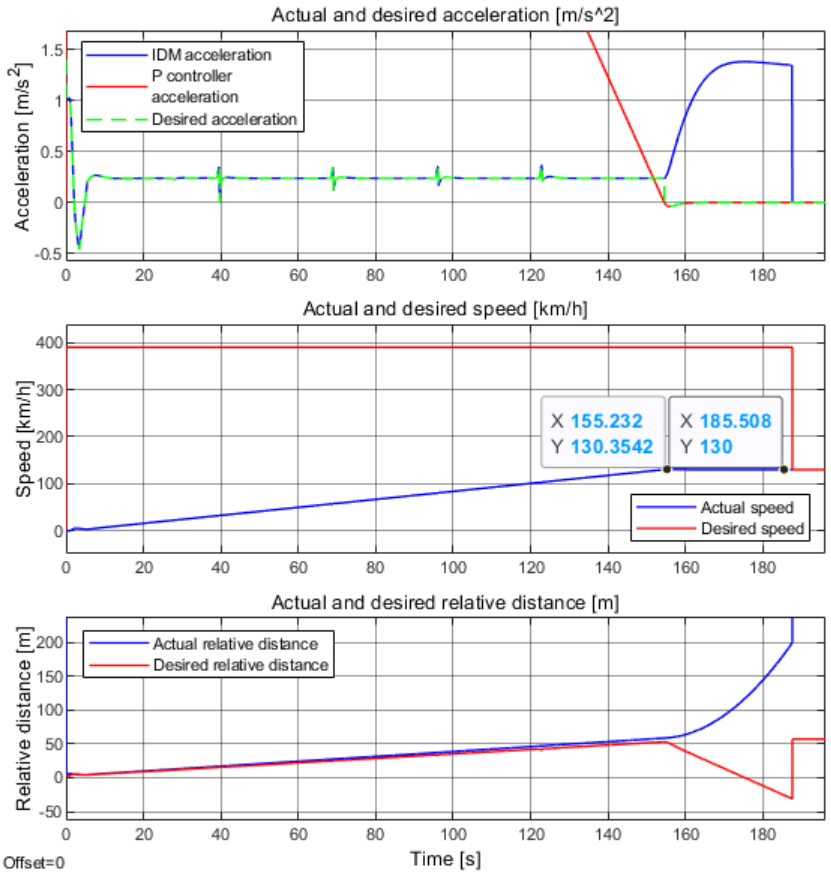

Figure 4-4: Car following scenario with the preceding vehicle exceeding the speed limit

As it can be seen, the ego car follows the desired relative distance correctly up until the ego car reaches the speed limit. This effect is a consequence of the controller switching strategy presented. In the acceleration plot, it can be seen that the desired acceleration

changes from the IDM acceleration to the P controller when the speed limit is crossed. This switching comes from the relative velocity being positive when the speed limit is surpassed, thus, indicating that the preceding car is developing speeds beyond the limit. This switching not only ensures that the ego car stops accelerating, but it also allows it to safely return to the speed limit, as seen in the actual speed plot.

4.1.4 Cut-in

The cut-in test consists in a traffic vehicle moving from an adjacent lane to the front of the ego vehicle. In particular, one traffic vehicle will be driving in the left lane with respect to the ego vehicle at 30 m from it. Then, the traffic vehicle will proceed to do a lane change towards the front of the ego vehicle, forcing it to brake to keep the safe distance from it.

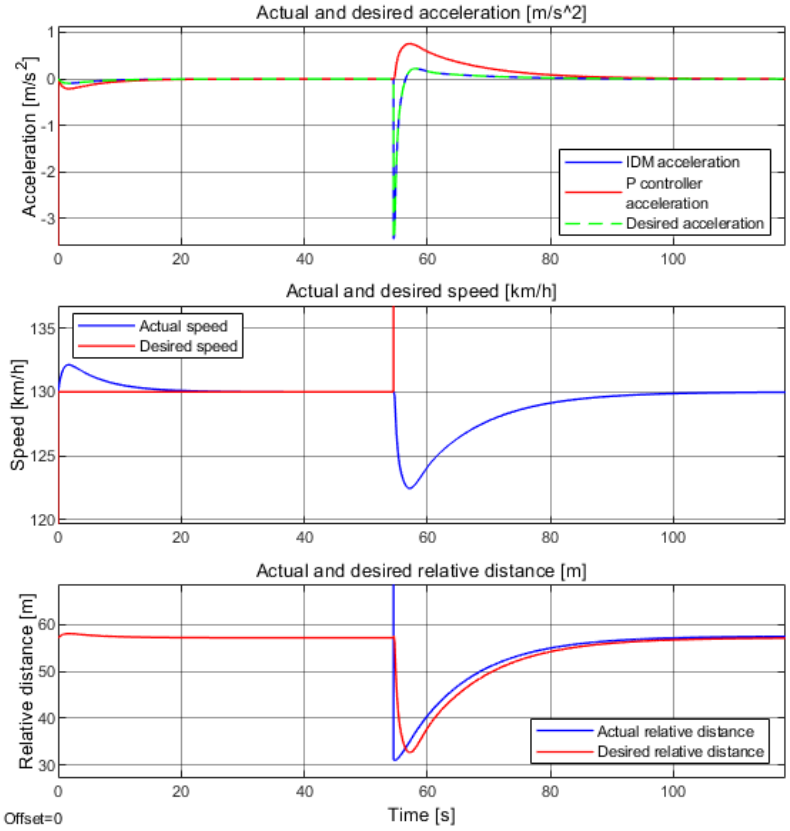

Figure 4-5: Cut In scenario at 130 km/h and 30m of relative distance

Analysing the relative distance plot, it can be seen that whenever the car is going in free flow at 130 km/h the desired relative distance is almost 60 m. This is why, when the preceding car cuts in at 30 m, the IDM generates a big decelerating response. In fact, if the distance with the preceding car in the cut-in scenario decreases with respect to the

desired relative distance in free flow, the deceleration response increases significantly, as expected. In addition, notice that a few seconds after the deceleration spike, there is an acceleration rebound that achieves the desired speed at the desired relative distance.

4.1.5 Cut-out

Similar to the cut-in scenario, this case consists in a traffic vehicle in front of the ego vehicle moving to another lane. If the car following speed is less than the road speed limit, an increase in velocity is expected when the traffic vehicle moves to the adjacent lane. In this case, to observe this behaviour, the car following speed will be of 90 km/h and the ego vehicle is expected to develop the road maximum legal speed of 130 km/h whenever the traffic vehicle changes lanes.

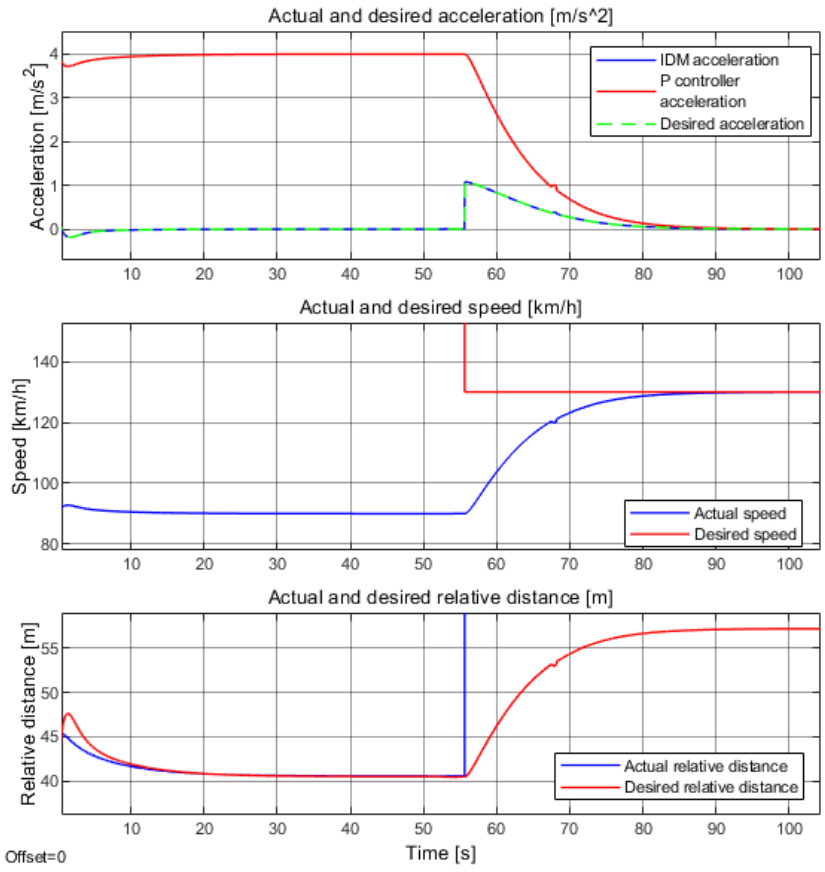

Figure 4-6: Cut-out scenario

As expected, the initial car following scenario makes the ego vehicle reach 90 km/h at the desired safe distance. Approximately at 55s, the preceding vehicle moves to another lane and the vehicle behaves precisely the same as in the free flow test case proposed

previously. As the preceding vehicle goes out of the ego vehicle's lane, the speed asymptotically reaches the road 130 km/h limit.

4.2 Lane keeping assist results

To assess lane keeping capabilities of the controller, different road profiles will be tested while monitoring the yaw error and the lateral deviation from the lane's centreline at the vehicle's centre of gravity (COG). The yaw error is calculated as follows

$$\psi_{error} = \psi_{car}^{COG} - \psi_{road}^{COG} \quad \text{(Eq 4-1)}$$

Where ψ_{car}^{COG} is the yaw angle of the car at the COG. ψ_{road}^{COG} is the road's yaw angle at a longitudinal distance so that the line between the COG and this point is perpendicular to the road.

Since CarMaker only provides the yaw angle of the car with respect to a fixed frame of reference, the road yaw angle at the COG is manually calculated. To do so, two points longitudinally close to each other are placed in the lane markings and the distance increments are compared to compute the tangent of the road yaw angle. As stated previously, since the car's yaw angle is expressed in a fixed frame of reference, the position increments between the two points can be calculated from the difference of each point's distance to that frame of reference's origin.

$$\tan\left(\psi_{road}^{COG}\right) = \frac{\Delta Y}{\Delta X} = \frac{Y_2 - Y_1}{X_2 - X_1} \quad \text{(Eq 4-2)}$$

Where $X_1,\ Y_1$ are the absolute coordinates of the point that intersects the lane marking with the perpendicular line to the marking that passes through the car's COG. Consequently, $X_2,\ Y_2$ are the absolute coordinates of the point placed directly in front of point 1.

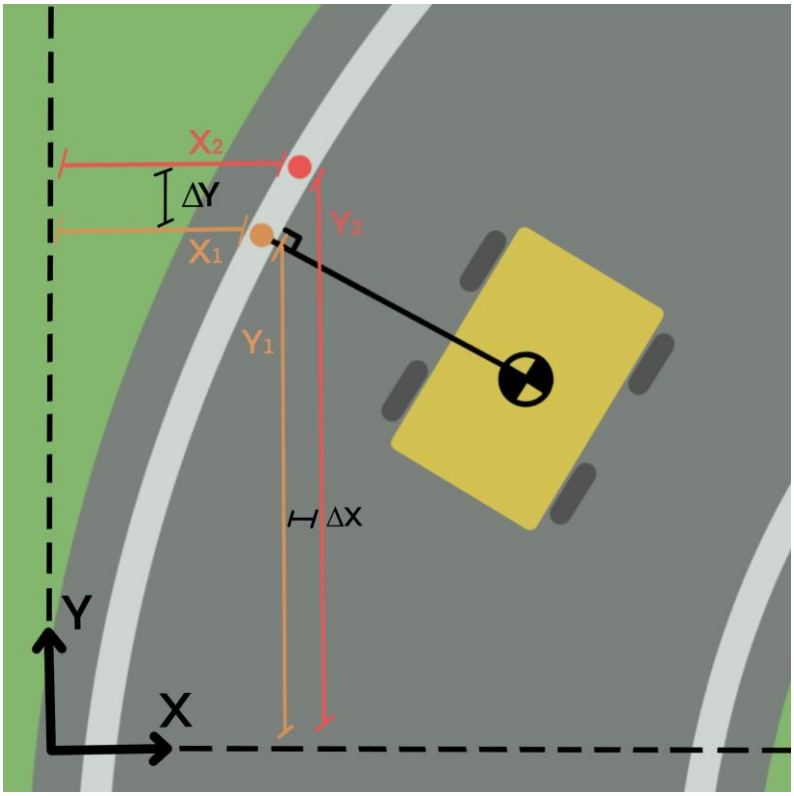

Figure 4-7: Road yaw angle calculation at the COG

To compute the lateral deviation from the centre line, the distance from the COG to the first point used for the yaw angle, (X_1, Y_1), is calculated, $d_{\substack{COG\ to\ lane \\ markings}}$, and compared to the length of half a lane, $l_{Half\ Lane}$. Expressed as a formula, the lateral deviation from the centreline is as follows:

$$\delta_{error} = d_{\substack{COG\ to\ lane \\ markings}} - l_{Half\ Lane} \qquad \text{(Eq 4-3)}$$

Despite the apparent simplicity of this equation, since the distance to the COG depends on the roads yaw angle the expression gets somehow more complicated. This distance is calculated as follows:

$$d_{\substack{COG\ to\ lane \\ markings}} = \Delta X_{COG} \cdot \sin\left(\psi_{road}^{COG}\right) + \Delta Y_{COG} \cdot \cos\left(\psi_{road}^{COG}\right) \qquad \text{(Eq 4-4)}$$

Where $\Delta X_{COG}, \Delta Y_{COG}$ are the absolute distance increments between the COG and the first point to calculate the road yaw angle. This formula essentially rotates the fixed frame of reference so that it matches a road fixed frame of reference.

With this procedure, different road profiles are tested to see the behaviour of the two magnitudes proposed.

4.2.1 S-shape road profile

This test case has the following road profile:

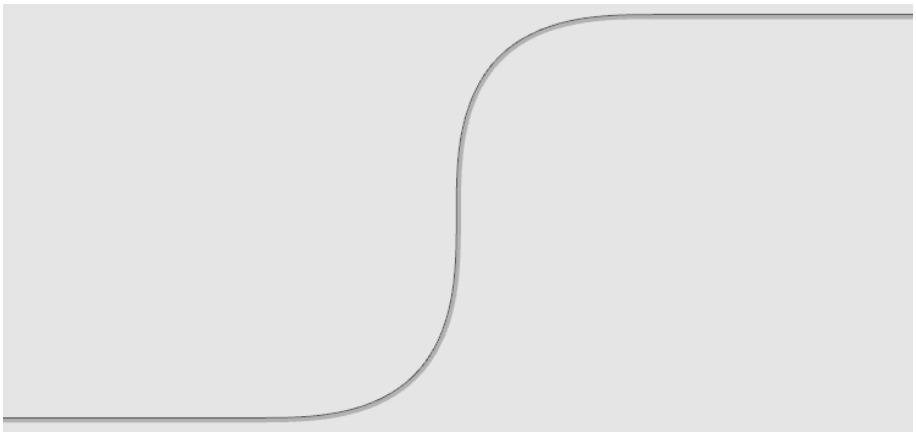

Figure 4-8: S-shape curve for LKA test

The starting point of the track is at the bottom left of the image. As it can be seen, first, the profile consists of a 500m straight road followed by a 90º counter clockwise turn. After that, a 90º clockwise turn is followed by another 500m straight segment. Note that the curvature of the turns is not constant. In fact, both 90º turns are composed of two 45º concatenated clothoid curves. These curves are characterised by their curvature increasing with their length. For the cases where the curves are initially connected to straight lines, they begin at a theoretical infinite radius (100000 in CarMaker) and decrease their radius up to 200m at 45º. If, on the contrary, the curves initially connect with the end of the previous 45º curves described, the radius goes from 200m to the theoretical infinite at 90º of the complete turn.

If the lateral deviation is monitored with the ego car's acceleration and speed:

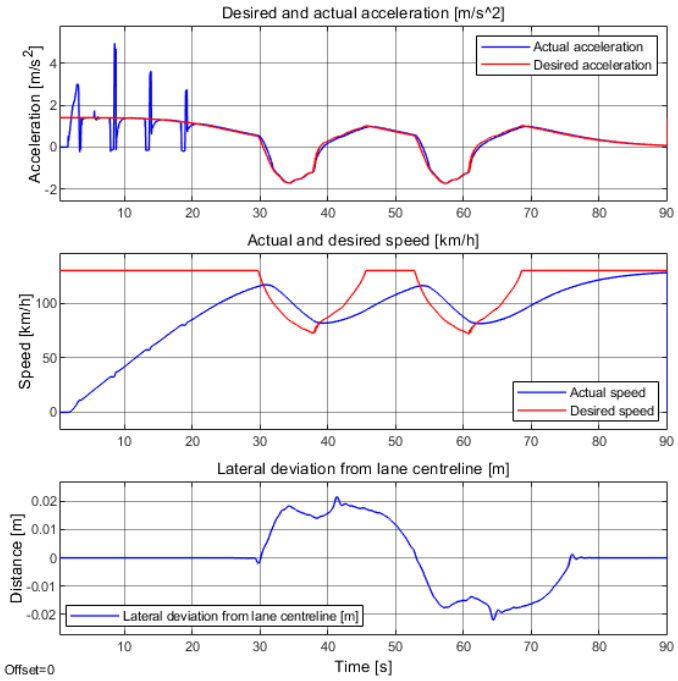

Figure 4-9: Acceleration, speed and lateral deviation of an S-shape test case

As it can be seen, the lateral deviation from the lane centreline lays on the centimetre spectrum, which, in terms of lane keeping assist control, is a very good sign for the controller lane tracking properties. Also notice how the increasing curvature sets the maximum velocity under the speed limit. This effect is caused by the curve approach method implemented monitoring the path's curvature.

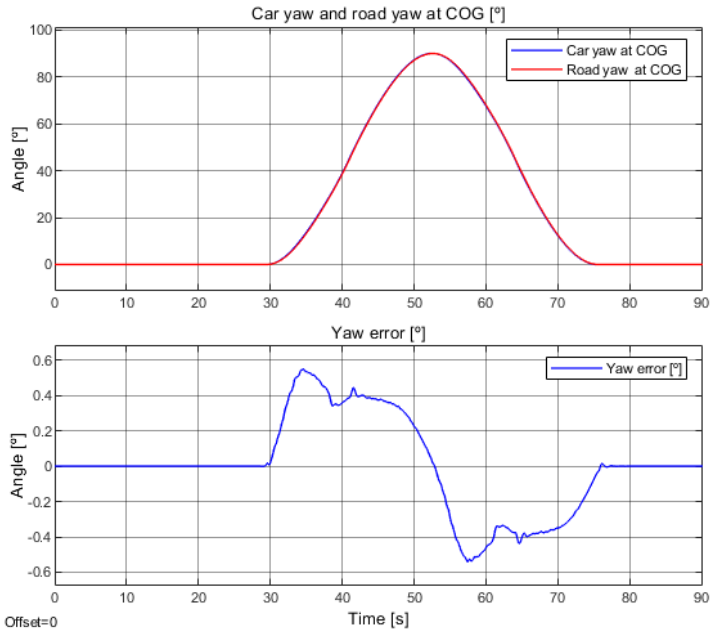

Figure 4-10: Comparison between car and road yaw, yaw error from an S-shape curve

Notice how the yaw angle of both the car and the road increase up to 90º and then decrease back to zero denoting the s-shape profile of the road. Also, note that the yaw error at the COG lays between plus and minus 0.6º which implies that the car is tightly oriented with the road.

4.2.2 Real test scenarios

To test the validity and the properties of the LKA controller in front of general road profile configurations, the algorithm is tested against real life scenarios at different speed limits. To do so, road profiles are imported from Google Maps into IPG Carmaker to ensure the maximum similarity between the simulated road profile to the actual road.

For these test cases, four real road profiles are proposed, two from Spanish roads and two from the United Kingdom. These specific road profiles have been selected due to their varying properties in both curvature and speed limits.

4.2.3 A-62 road from Cavia to Quintanapalla (Spain)

This 28 km long road profile displayed relatively high curvature at highway speeds, which provide the perfect scenario to test the LKA controller's high speed behaviour. The profile has the following shape:

Figure 4-11: Real life scenario from Cavia to Quintanapalla (Spain)

Even though it may not look especially curvy, this is a highway road and, thus, the lateral acceleration due to these radii can be significant, as the velocities can reach high values. In this case, the road has been tested with a speed limit of 120 km/h, following the Spanish driving law for highways, and the results extracted from it are the following:

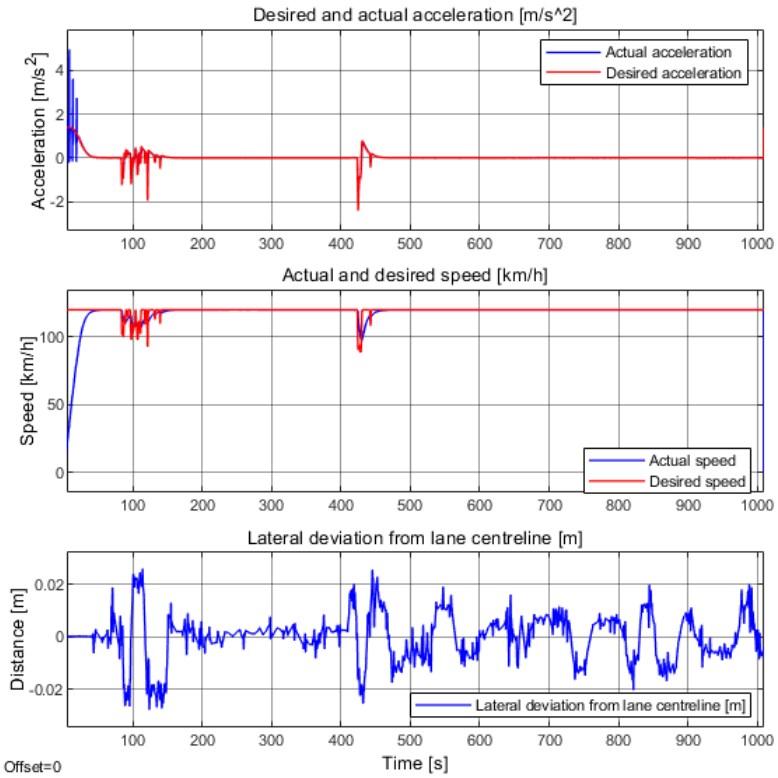

Figure 4-12: Acceleration, speed and lateral deviation of a Spanish highway

Notice how the vehicle barely needs to brake due to the curvature of the road despite the noticeable curvature of the road. In fact, notice that the desired speed drops close 100 km/h twice, denoting that the maximum curvature of this test is less than the S-shape case. Also, note that the lateral deviation signal is always in between plus and minus 2 cm, approximately. As in the S-shape case, this results in the ego vehicle tightly tracking the lane centreline.

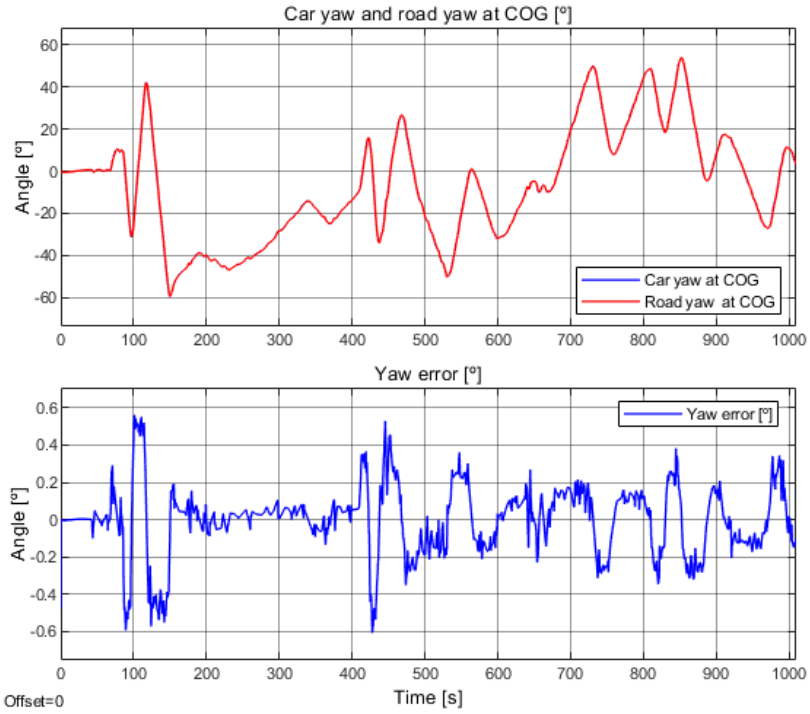

Figure 4-13: Yaw error from a Spanish highway

As it can be seen, the peaks of error occur at the points where the curvature is high. In Figure 4-13, these regions can be identified by sharp peaks and valleys of the car and road yaw angles. Essentially, a sudden change of yaw angle implies that the curvature is high and vehicle dynamics make it harder for the controller to achieve zero yaw error.

4.2.4 Arrabassada road from Barcelona to Tibidabo (Spain)

This road is one of the curviest roads of Barcelona at a speed limit of 50 km/h. Hence, it will not only test the response to tight curves but also the response of the controller at a moderate velocity. The road profile looks like the following:

Figure 4-14: Arrabassada road profile (Spain)

The results yielded by this test are the following:

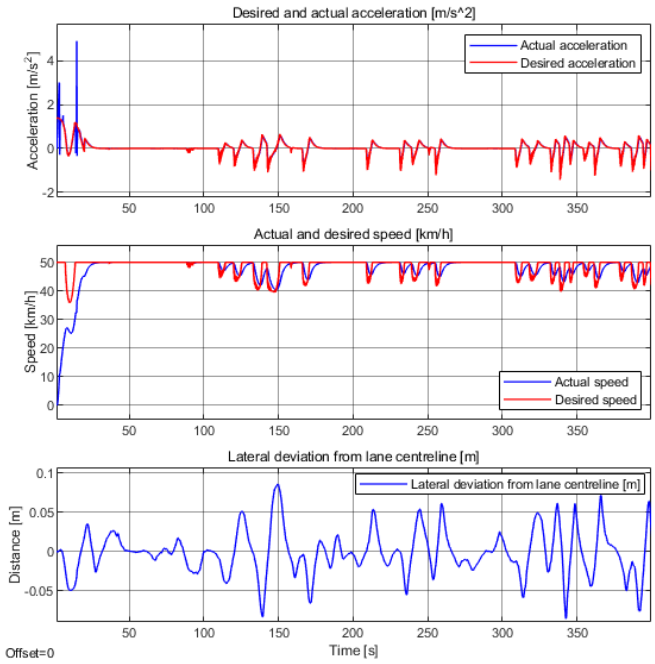

Figure 4-15: Acceleration, speed and lateral deviation of Arrabassada road (Spain)

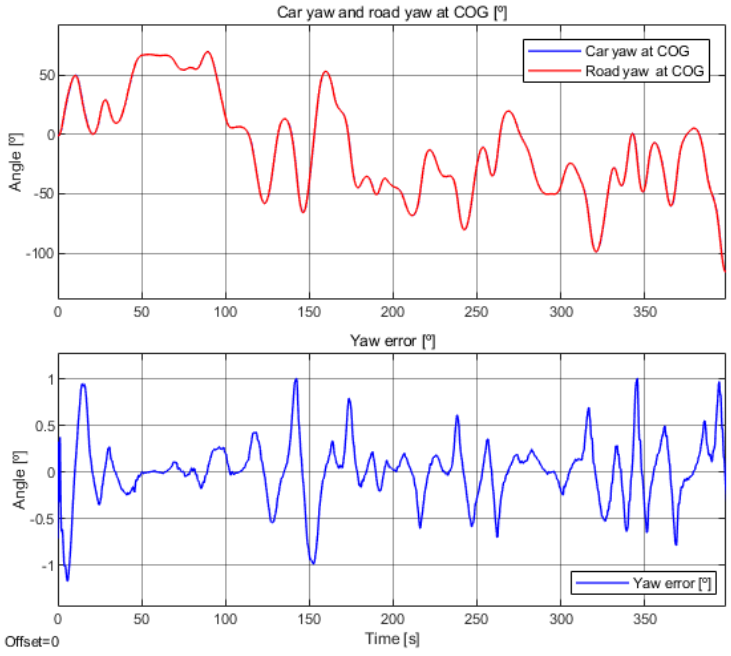

Figure 4-16: Yaw error of Arrabassada road profile

Notice how, now, due to high curvatures, even at moderate speeds, the lateral deviation and the yaw error increase with respect to the previous test cases. As previously stated, tight curves generate vehicle dynamics that try to force the vehicle outwards from the

curve and, hence, the LKA controller has more trouble tracking the lane centreline at the correct orientation. Nevertheless, the maximum amplitude increase with respect to the previous case in lateral deviation is about 5 cm and 0.4º for yaw error, which are considered acceptable.

4.2.5 A4086 single carriageway road from Wales (UK)

An even curvier road profile is proposed with a higher speed limit than the previous as it is a UK single carriageway (60 mph = 96 km/h). Consequently, this means that the desired speed will need to decrease sharply and more frequently to approach curves safely. The road profile has the following shape:

Figure 4-17: A4086 road profile from Wales (UK)

And the results yielded by the test are:

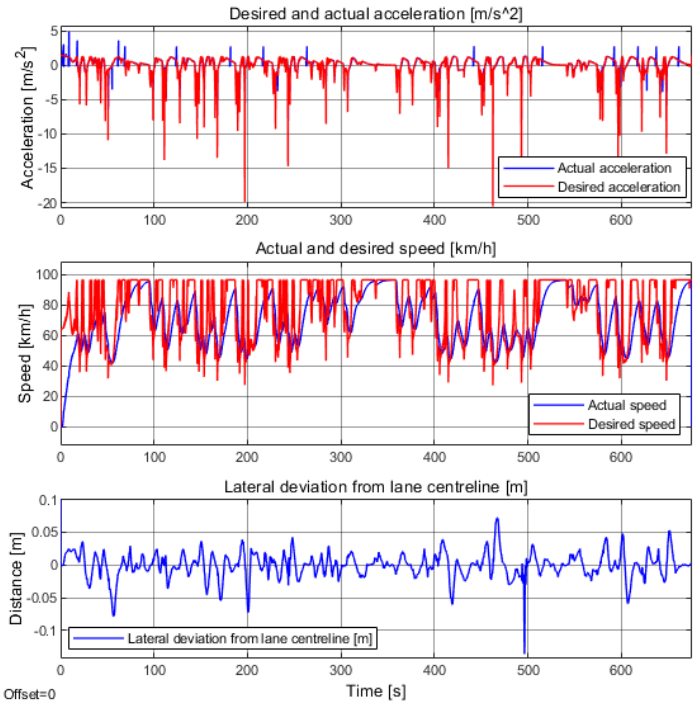

Figure 4-18: Acceleration, speed and lateral deviation of A4086 (Wales)

As expected, due to the speed limit being much higher than the one at Arrabassada road but having relatively similar curves, the desired speed needs to drop to similar speed

levels when a curve is starting, but now approaching it with a much higher starting speed. In turn, notice the big deceleration spikes generated by big sudden curvature changes to achieve the desired speeds commanded. Notice how these create the biggest lateral deviation errors. This effect is a consequence of the vehicle tyres slipping and loosing traction. To address this issue, a jerk planner would be beneficial to avoid sudden deceleration spikes and generate smoother acceleration commands, avoiding tyre slipping. A similar effect happens with the yaw error at the centre of gravity, as Figure 4-19 shows.

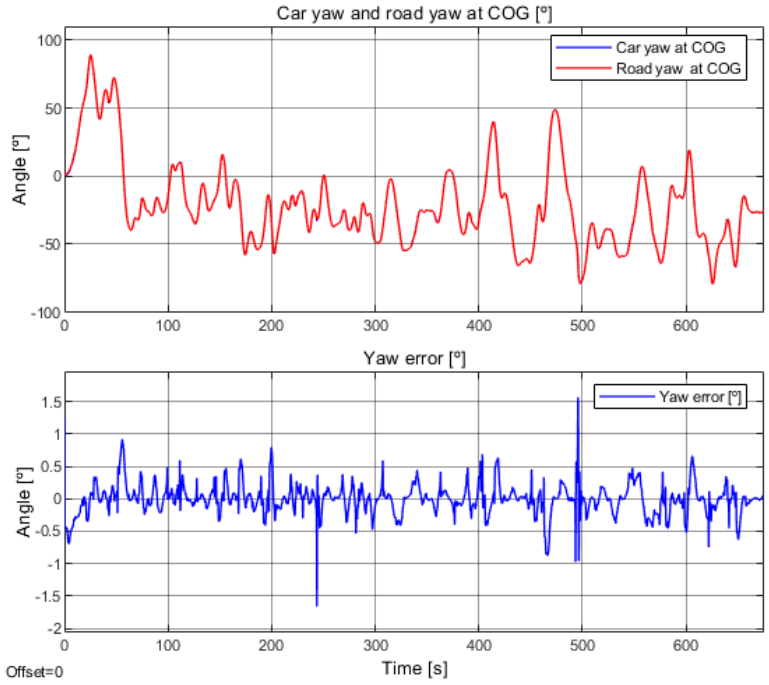

Figure 4-19: Yaw error of A4086 (Wales)

4.2.6 M1 motorway road from England (UK)

Finally, probably the most important road from the UK is tested between Milton Keynes and the outskirts of London. Even though this test case is not especially curvy, it will test the endurance and reliability of the algorithm in long distance scenarios, with a length of roughly 65 km. The profile is:

Figure 4-20: M1 road profile between Milton Keynes and London

The results are as follows:

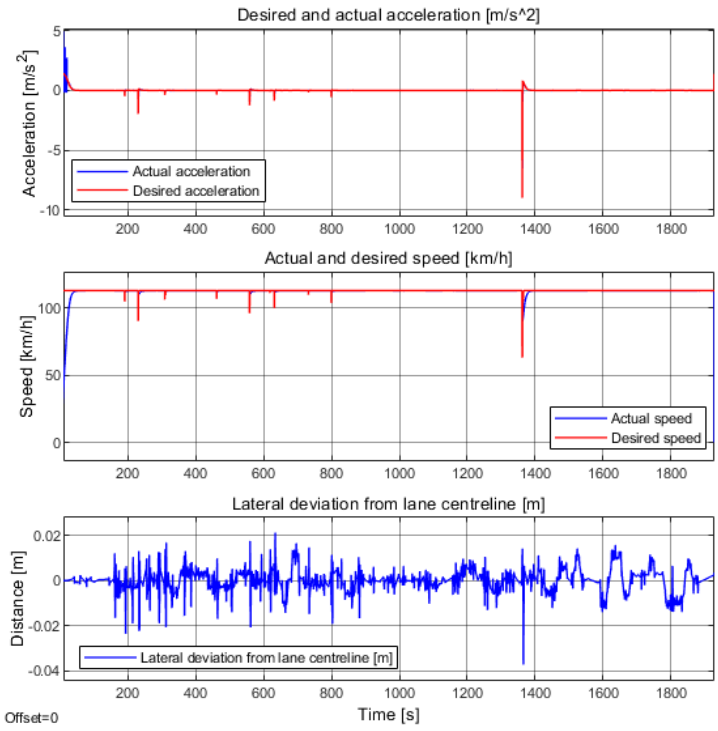

Figure 4-21: Acceleration, speed and lateral deviation of M1 motorway (UK)

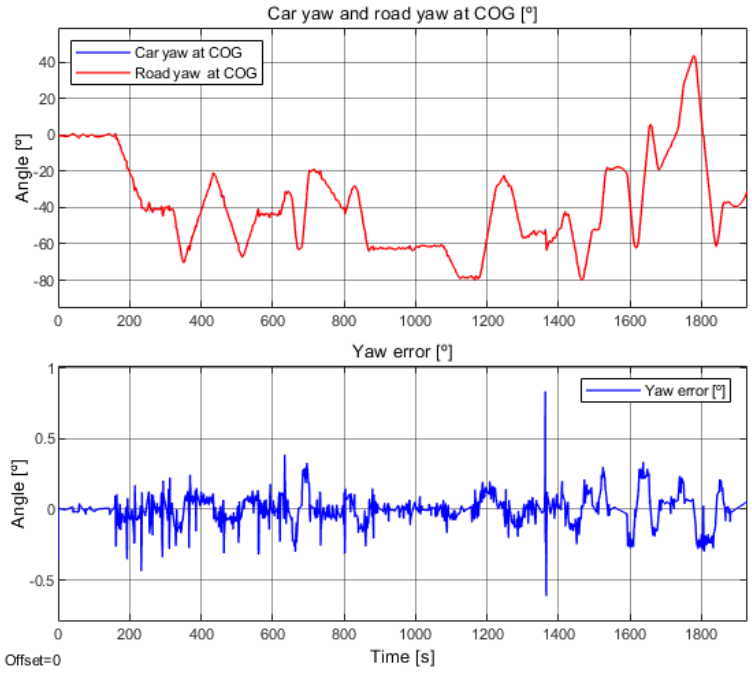

Figure 4-22: Yaw error from M1 motorway (UK)

Notice that, due to the long tested time, the few decreases of speed that are required look like small drop spikes in velocity. Nevertheless, the deceleration behaviour is the same as in the previous test cases showed. Again, notice how the big deceleration spike, at around 1350s, clearly generates the biggest position difference between the car and the lane centreline due to tyre slipping at a curve. A similar effect can be seen in the yaw angle. Despite that, the maximum values are between the desired ranges, both in yaw and lateral deviation.

4.3 Combined ACC and LKA results

To assess the combined ACC and LKA controller properties, a general test case is proposed. This test case consists in a portion of the previously analysed M1 motorway with stochastic traffic generated by CarMaker. The road profile analysed is the following:

Figure 4-23: Shorter M1 motorway track for testing with stochastic traffic

The stochastic traffic is composed not only by cars, but also motorcycles, trucks and semi-trucks. Thus, since the generated traffic follows the specified country speed limits, if the traffic is restricted to a single lane, the maximum speed of the traffic is limited by the slowest one, in this case, trucks and semi-trucks.

Due to this effect, the traffic is set to change lanes so that the ego vehicle can perform all ACC manoeuvres, specially cut-in and cut out scenarios. In particular, the traffic will begin in the same lane as the ego vehicle and along the route five lane changes will take place, three from the ego vehicle's lane to an adjacent and two from an adjacent to the ego vehicle's lane. An odd number of lane changes is chosen so that the ego vehicle is not obstructed by any traffic at the end segment of the test case and ends the track in a free flow state.

The results from this test case are the following:

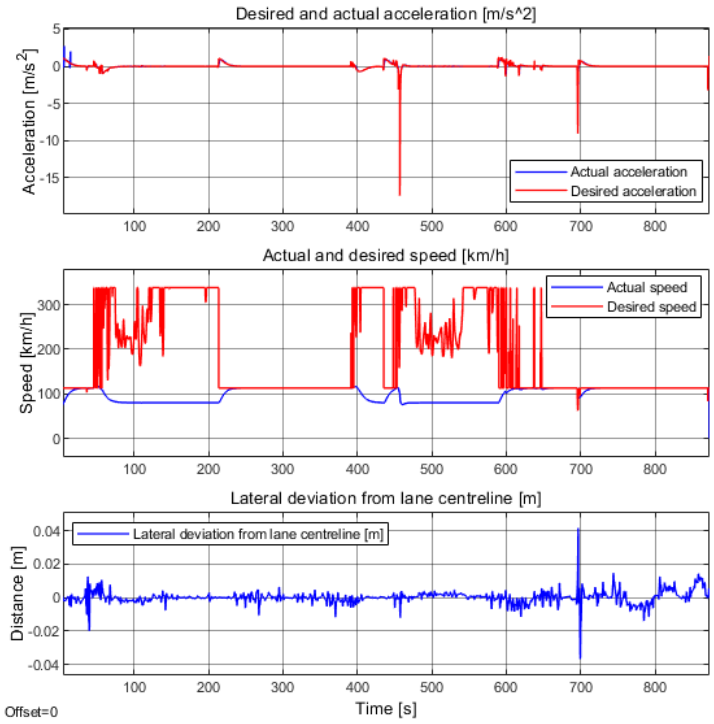

Figure 4-24: Acceleration, speed and lateral deviation of M1 motorway (UK) with stochastic traffic

As a reminder, the desired speed becomes three times the speed limit whenever there is a vehicle in the radar scope of the ego vehicle. Hence, whenever there is a raising edge on the desired speed curve, a preceding vehicle has entered the radar sensor range. In fact, notice how raising edges of desired speed imply negative accelerations, thus, braking, while falling edges generate a positive one, usually in cut-out scenarios.

Nevertheless, notice that in two instances, approximately at 450s and 700s, the desired acceleration spikes in a sudden braking manoeuvre. Also, see that the first one occurs in a desired speed raising edge, while in the second one the desired speed decreases. These effects denote that the first instance is generated due to a cut-in scenario while the second one is just caused by a sudden change of road curvature.

Despite the big deceleration spikes, only the second seems to generate a major disturbance on the lateral deviation and yaw error, as seen also in Figure 4-25. This effect comes from the cut-in scenario taking place in a mostly straight segment of the road. Thus, even if the big deceleration spike causes the ego vehicle to slip, the loss of traction barely affects both magnitudes. If the same cut-in scenario would have taken place on a curved road profile, the yaw and lateral deviation spikes would have been similar to the ones generated by the sudden big curvature change.

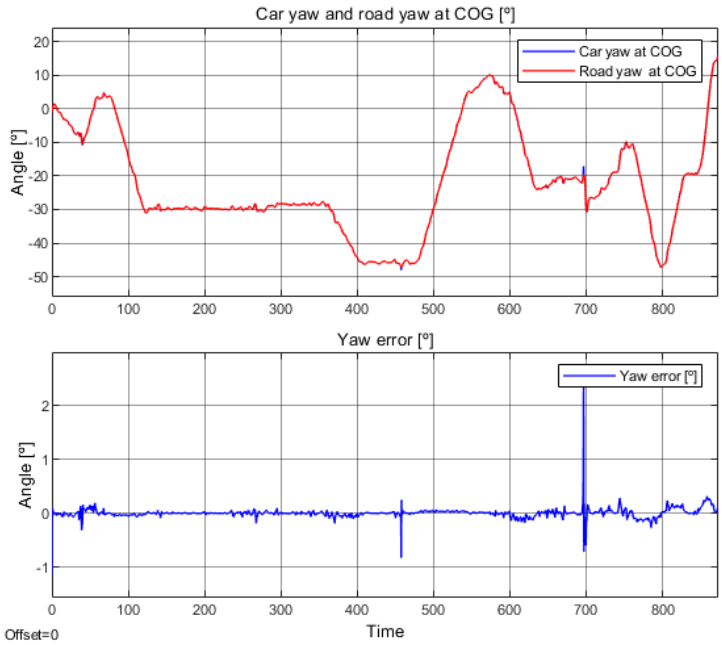

Figure 4-25: Yaw error of the M1 motorway (UK) with stochastic traffic

Finally, to see if the ACC correctly tracks the desired relative distance upon facing the different scenarios presented, the actual and desired relative distances between the ego vehicle and the preceding vehicle are plotted.

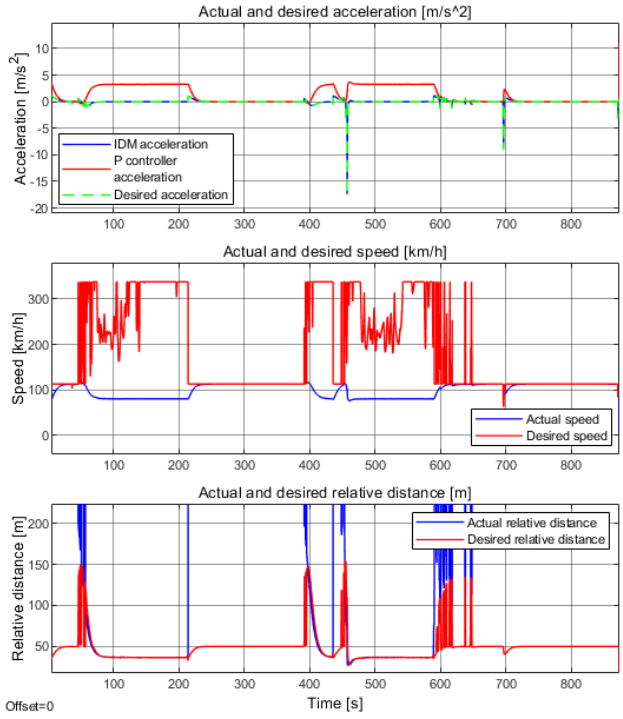

Figure 4-26: Acceleration, speed and relative distance evolution in a general test case with stochastic traffic

Here, it is even clearer that whenever the desired speed raises to three times the speed limit, there is a traffic object in the radar scope as the actual relative distance drops from a big value (theoretical infinite) to the measured value. Also, the plot reveals that when a traffic object is in the sensor’s scope, relative distance is tightly tracked, especially in steady state conditions, such as car-following scenarios.

5 Conclusion

This project covered the development and implementation of a longitudinal and lateral controller for an autonomous vehicle that includes adaptive cruise control and lane keeping assist. The hole body of work has been coded/modelled using Matlab/Simulink and simulated using IPG Carmaker. The vehicle simulated has a radar sensor that not only provides information about the presence of a preceding object but also relevant magnitudes such as the relative distance and speed. Also, a line sensor is fitted into the vehicle's windshield rear-view mirror to extract data from the lane bounds.

To have a reference from previous research, and to have a solid basis from where to commence the development, related concepts and control strategies have been gathered from a topic specific literature review. This literature consolidated the basis of the project as many of the control strategies' core behaviour is extracted directly from literature. In addition, it has provided a sense of the state of the art of the public lateral and longitudinal controllers developed.

Concerning the development of the algorithm, first, the adaptive cruise controller (ACC) has been developed based on the intelligent driver model. Due to the fact that this model losses distance tracking capabilities the closer the ego car goes to the desired speed, an algorithm is developed around it to enhance it. This algorithm consists in increasing the desired speed to a much higher value than the speed limit whenever an object is in the ACC sensor scope. Consequently, the relative speed beyond the speed limit needs to be tracked to decide whether to proceed tracking the relative distance between vehicles or return to the maximum legal speed, if exceeded.

On the other hand, lane keeping assist (LKA) has been developed by monitoring not only the lateral deviation from the lane centreline, but also the yaw error between the car and the path followed. To do so, Catmull-Rom curves have been fitted into the lane centreline and both yaw and position are monitored in a velocity dependent point of the curve. This moving point ensures that the faster the vehicle goes, the farther ahead the point is and, thus, more steering predictive behaviour is added to the control. By comparing these two magnitudes with their respective ones from the ego vehicle, an error can be extracted and passed through a PID controller for each, generating the steering commands needed.

The last step of development was to join the two functionalities and make them work safely. Initial work showed no interferences between both functionalities but, since now

the acceleration was fully commanded by the ACC, a method to approach curves is developed. This consists in monitoring the curvature of the fitted Catmull-Rom curve on a completely independent speed dependent point. By using this innovative method, the desired speed of the intelligent driver model is modified to be inversely proportional to the road curvature. The fact that the moving point is independent from the LKA one is because speed control needs more look ahead distance than steering control to react.

In addition, to make the project safer and consider the sensor configuration of the simulated vehicle, a safety measure is added to the control strategy. This feature monitors the intersection of the fitted path and the radar sensor cone whenever a traffic object disappears from the sensor scope. This ensures that when a preceding vehicle is taking a curve and escapes the sensor due to a high curvature and a low horizontal aperture angle, the ego vehicle adapts its acceleration to expect an object entering through the centre of the lane at the edge of the sensor.

Finally, to test the controller and its properties against different scenarios, not only the combined longitudinal and lateral controller is assessed, but also the ACC and LKA individually. This procedure ensures that all test cases not covered in possible general scenarios are evaluated and considered.

First, an analysis on the ACC feature is done where the desired and actual speed and relative distance are monitored. This testing provides insight about the speed and distance tracking capabilities of the ACC. In all instances, the magnitudes showed zero steady state error between the actual and the desired values.

Secondly, several tests for the LKA controller are done using synthetic and real life road profiles from Google Maps. The magnitudes evaluated are the yaw error and the lateral deviation from the lane centreline at the centre of gravity. Results showed that lateral deviation stayed below 5cm in the majority of the test cases and a yaw error below 1º. In terms of lane keeping, these magnitudes indicate a desired behaviour as in comparison to the width of the lane, for the case of the lateral deviation, is orders of magnitude smaller.

5.1 Future work

Even though the results are promising, the implementation needs to be polished further to achieve its full potential.

First and the most significant feature to add would be to pass the acceleration through a jerk manager to ensure passenger comfort and safe braking manoeuvres. Now, for instance, if a vehicle cuts-in in front of the ego vehicle at high speeds and small relative distance, the intelligent driver model reacts too forcefully, sometimes even causing tyre slipping. Implementing this feature would force the desired acceleration to be smoother and, thus, more comfortable and safe.

Another feature to consider would be to consider a more complex sensor configuration to broaden the capabilities of the controller. For instance, a radar sensor placed at the back bumper could prevent crashes from the rear and even help in implementing a lane change functionality. Also, a camera sensor at the front with a wider aperture angle than the radar could improve the safety capabilities, especially during tight curves and cut-in scenarios. With the latter configuration, the safety feature that considers the intersection of the sensor cone and the path would be required in fewer instances, thus, improving safety and reliability. In cut-in scenarios, the acceleration could start braking before the car enters the radar scope and increase the distance with the entering car.

In addition, to implement the acceleration switching strategy more robustly, the following logic could be implemented:

$$\dot{v} = \begin{cases} P\ controller, \ \ if\ car\ speed + rel.vel > limit \\ \\ Intelligent\ Driver\ Model, \ \ else \end{cases} \quad \text{(Eq 5-1)}$$

By using this logic, the acceleration controller would only switch the P controller to return to the speed limit whenever the preceding vehicle would go faster than the speed limit. Using the logic implemented in this project, there is a rare scenario in which the ego car would follow a vehicle beyond the speed limit. As explained previously, the ego car goes beyond the speed limit to achieve the desired relative distance proposed by the IDM even if the preceding vehicle has never exceeded it. If in that particular instant the preceding vehicle would increase its velocity just below the ego car's, the relative velocity would still be negative and the ego vehicle would increase its velocity even more to achieve the new desired relative distance. By following this escalating velocity method, the preceding vehicle could, theoretically, be able to drag the ego vehicle beyond the

speed limit and stabilize. This new logic, though, does not rely on both conditions being true and combines both into one. In fact, the expression of the new condition monitors the actual speed of the preceding vehicle, making the approach much more robust to ill-intentioned manoeuvres.

Finally, a camera sensor could be introduced with incorporated machine learning to predict the possible outcomes of a preceding traffic object and adapt both the steering and acceleration accordingly. Also, the type of traffic object could inform the controller of how to approach the manoeuvre. For instance, the ego vehicle should not act the same with a big boulder as with a beach ball. The first one is definitely a threat if not acted cautiously, but if reckless manoeuvres are done in the second case, a possible threat might appear, for example, in lane changes.

REFERENCES

[1] P. D. I. Torino, “ADAS virtual validation : ACC and AEB case study with IPG CarMaker,” 2018.

[2] O. Gietelink, *Design and Validation of Advanced Driver Assistance Systems*. .

[3] A. Kesting, M. Treiber, M. Scho, and D. Helbing, “Adaptive cruise control design for active congestion avoidance,” vol. 16, pp. 668–683, 2008.

[4] R. Detection, “Real Time Lane Detection and Tracking System Evaluated in a Hardware-in-The-Loop Simulator,” pp. 1336–1343, 2010.

[5] C. I. C. K. N. Spentzas, “A path-following driver model with longitudinal and lateral control of vehicle ’ s motion,” pp. 257–266, 2009.

[6] “uk.mathworks.com.” [Online]. Available: https://uk.mathworks.com/products/matlab.html. [Accessed: 19-Jul-2021].

[7] “Centripetal Catmull-Rom Spline.” [Online]. Available: https://en.wikipedia.org/wiki/Centripetal_Catmull–Rom_spline. [Accessed: 19-Jul-2021].

[8] V. Dynamics, “5 . Handling Modelling.”

APPENDICES

Appendix A Simulink models

A.1 ACC Subsystem

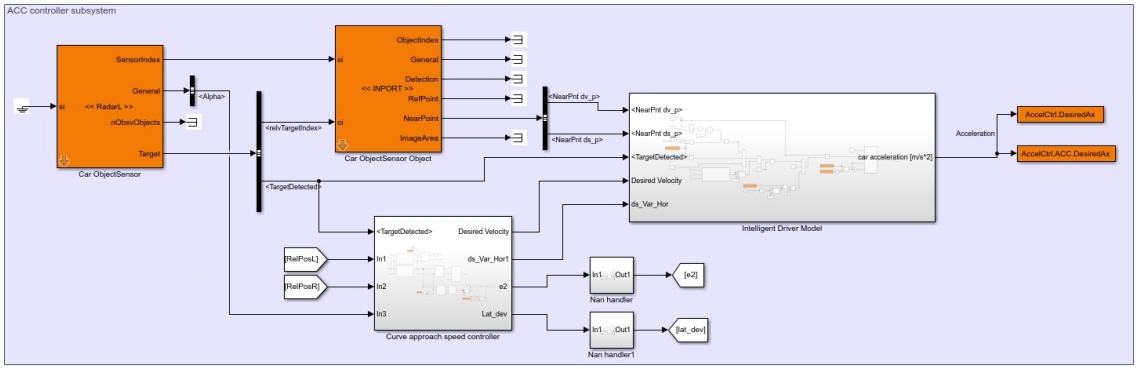

Figure A-1: ACC general subsystem

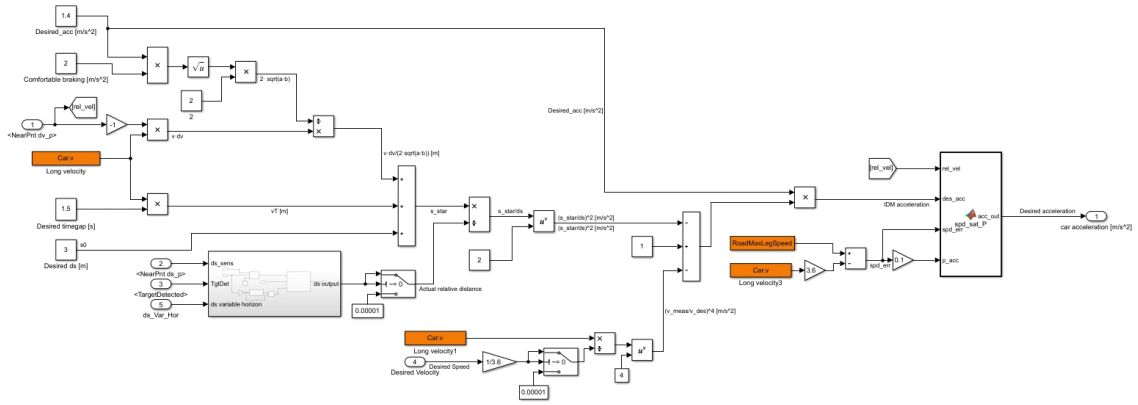

Figure A-2:Intelligent driver model subsystem detail

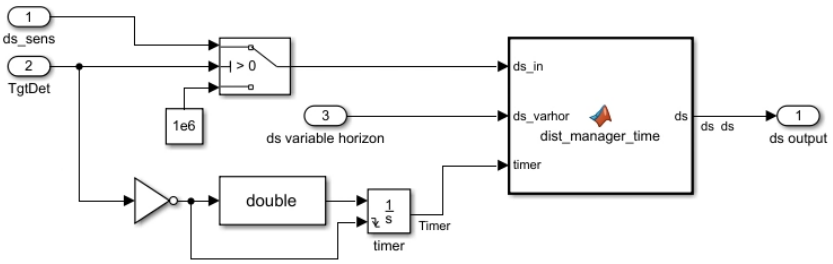

Figure A-3: Crash avoidance time manager subsystem

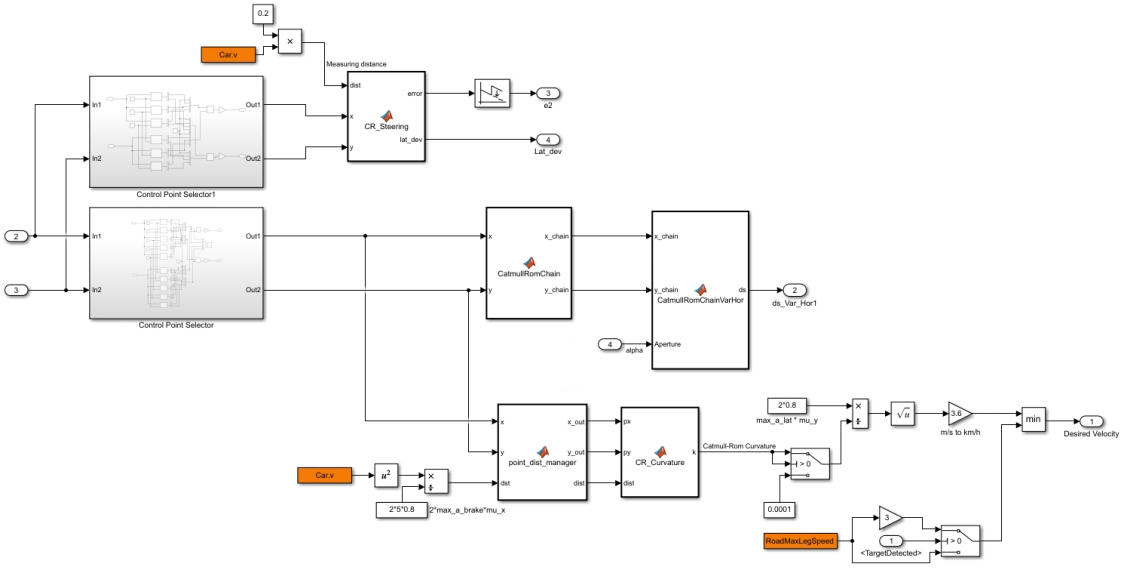

Figure A-4: Catmull-Rom curve calculation subsystem

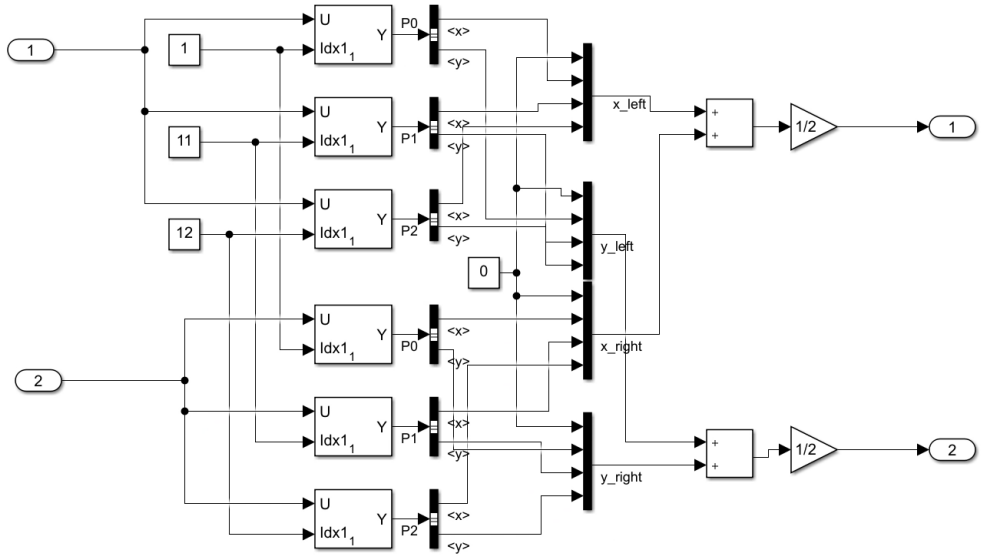

Figure A-5: Catmull-Rom fitted curves for steering

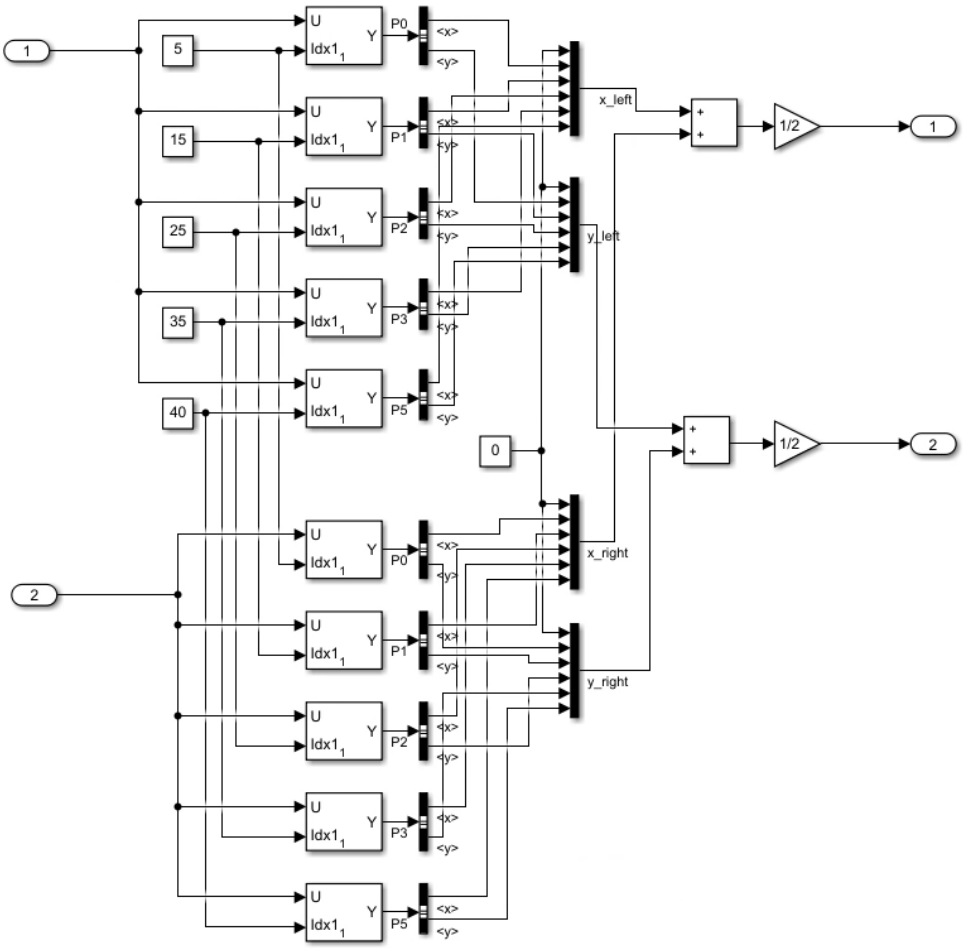

Figure A-6: Fitted Catmull-Rom curves for curvature monitoring and crash avoidance

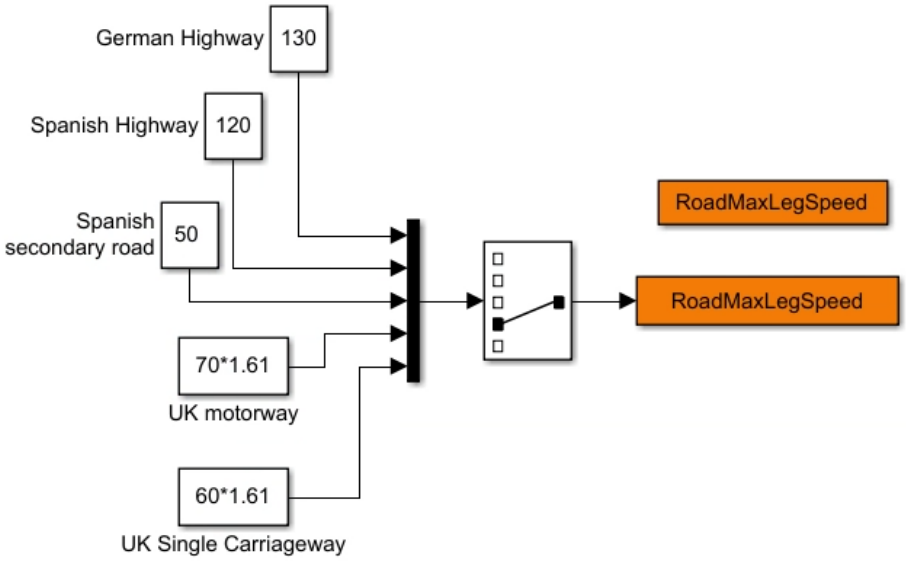

Figure A-7: Road maximum legal speed definition and management

A.2 LKA Subsystem

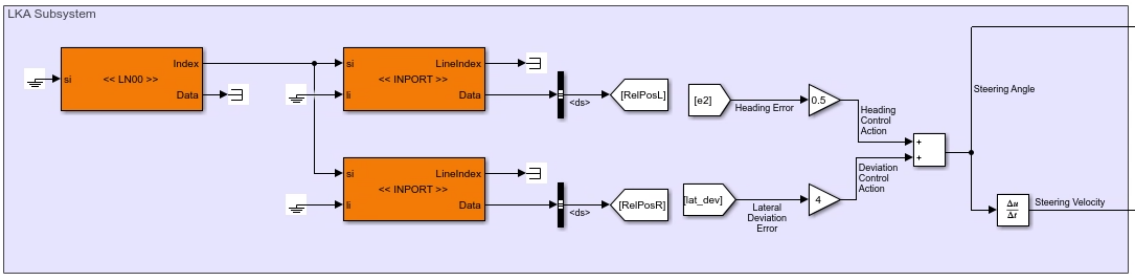

Figure A-8: LKA general subsystem

Appendix B Code developed

B.1 Catmull-Rom lateral deviation and yaw error calculation

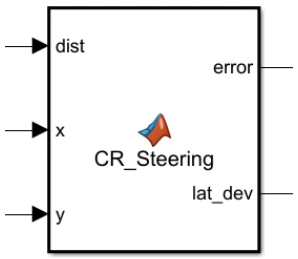

Figure B-1: Catmull-Rom block for calculating lateral deviation and yaw error

```
function [error, lat_dev]=  CR_Steering(dist, x, y)
    %Calculates lateral deviation and yaw error from a speed
    %depending moving point at dist from Catmull-Rom curves fitted to
    %the lane centreline
    px = x(1:4);
    py = y(1:4);
    alphaCR = 0.5;

    t0 = 0;
    t1 = (sqrt((px(2)-px(1))^2+(py(2)-py(1))^2))^alphaCR + t0;
    t2 = (sqrt((px(3)-px(2))^2+(py(3)-py(2))^2))^alphaCR + t1;
    t3 = (sqrt((px(4)-px(3))^2+(py(4)-py(3))^2))^alphaCR + t2;
    if dist < px(2)
        t = t1;
    elseif dist > px(3)
        t = t2;
    else
        t = ((t2-t1)/(px(3)-px(2)))*(dist-px(2)) + t1;
    end
    %% Lateral deviation extraction
    A1x = ((t1-t)/(t1-t0))*px(1) + ((t-t0)/(t1-t0))*px(2);
    A1y = ((t1-t)/(t1-t0))*py(1) + ((t-t0)/(t1-t0))*py(2);

    A2x = ((t2-t)/(t2-t1))*px(2) + ((t-t1)/(t2-t1))*px(3);
```

```
    A2y = ((t2-t)/(t2-t1))*py(2) + ((t-t1)/(t2-t1))*py(3);

    A3x = ((t3-t)/(t3-t2))*px(3) + ((t-t2)/(t3-t2))*px(4);
    A3y = ((t3-t)/(t3-t2))*py(3) + ((t-t2)/(t3-t2))*py(4);

    B1x = ((t2-t)/(t2-t0))*A1x + ((t-t0)/(t2-t0))*A2x;
    B1y = ((t2-t)/(t2-t0))*A1y + ((t-t0)/(t2-t0))*A2y;

    B2x = ((t3-t)/(t3-t1))*A2x + ((t-t1)/(t3-t1))*A3x;
    B2y = ((t3-t)/(t3-t1))*A2y + ((t-t1)/(t3-t1))*A3y;

    lat_dev = ((t2-t)/(t2-t1))*B1y + ((t-t1)/(t2-t1))*B2y;

    %% Yaw error calculation
    A1xp = (1/(t1-t0))*(px(2)-px(1));
    A1yp = (1/(t1-t0))*(py(2)-py(1));

    A2xp = (1/(t2-t1))*(px(3)-px(2));
    A2yp = (1/(t2-t1))*(py(3)-py(2));

    A3xp = (1/(t3-t2))*(px(4)-px(3));
    A3yp = (1/(t3-t2))*(py(4)-py(3));

    B1xp = (1/(t2-t0))*(A2x-A1x+(t2-t)*A1xp+(t-t0)*A2xp);
    B1yp = (1/(t2-t0))*(A2y-A1y+(t2-t)*A1yp+(t-t0)*A2yp);

    B2xp = (1/(t3-t1))*(A3x-A2x+(t3-t)*A2xp+(t-t1)*A3xp);
    B2yp = (1/(t3-t1))*(A3y-A2y+(t3-t)*A2yp+(t-t1)*A3yp);

    Cxp = (1/(t2-t1))*(B2x-B1x+(t2-t)*B1xp+(t-t1)*B2xp);
    Cyp = (1/(t2-t1))*(B2y-B1y+(t2-t)*B1yp+(t-t1)*B2yp);

    error = atan2(Cyp,Cxp);

end
```

B.2 Multiple Catmull-Rom chain point generator

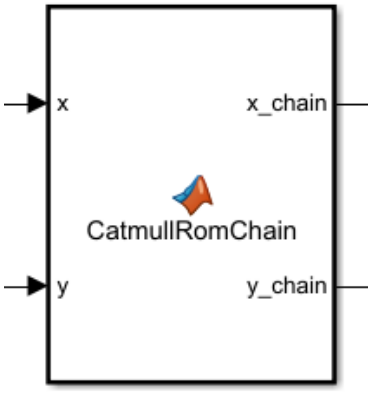

Figure B-2: Multiple Catmull-Rom chain point generator

```
function [x_chain, y_chain] =  CatmullRomChain(x, y)
    %Chains together all points generated by the different Catmull-Rom
    %splines generated with consecutive control points
    [x1, y1] = CatmullRomSpline(x(1:4), y(1:4), 100);
    [x2, y2] = CatmullRomSpline(x(2:5), y(2:5), 100);
    [x3, y3] = CatmullRomSpline(x(3:6), y(3:6), 100);
    x_chain = [x1, x2, x3];
    y_chain = [y1, y2, y3];

function [x_cat, y_cat] = CatmullRomSpline(px, py, npoints)
    % Defines a Catmull-Rom spline with four control points with
    %(px,py) coordinates and generates n points of this spline segment
    %(npoints)
    alphaCR = 0.5;
    t0 = 0;
    t1 = (sqrt((px(2)-px(1))^2+(py(2)-py(1))^2))^alphaCR + t0;
    t2 = (sqrt((px(3)-px(2))^2+(py(3)-py(2))^2))^alphaCR + t1;
    t3 = (sqrt((px(4)-px(3))^2+(py(4)-py(3))^2))^alphaCR + t2;
    x_cat = zeros(1, npoints);
    y_cat = zeros(1, npoints);
    t_vect = linspace(t1, t2, npoints);
    for i = 1:length(t_vect)
        t = t_vect(i);
        %%
        A1x = ((t1-t)/(t1-t0))*px(1) + ((t-t0)/(t1-t0))*px(2);
        A1y = ((t1-t)/(t1-t0))*py(1) + ((t-t0)/(t1-t0))*py(2);

        A2x = ((t2-t)/(t2-t1))*px(2) + ((t-t1)/(t2-t1))*px(3);
        A2y = ((t2-t)/(t2-t1))*py(2) + ((t-t1)/(t2-t1))*py(3);

        A3x = ((t3-t)/(t3-t2))*px(3) + ((t-t2)/(t3-t2))*px(4);
        A3y = ((t3-t)/(t3-t2))*py(3) + ((t-t2)/(t3-t2))*py(4);

        B1x = ((t2-t)/(t2-t0))*A1x + ((t-t0)/(t2-t0))*A2x;
        B1y = ((t2-t)/(t2-t0))*A1y + ((t-t0)/(t2-t0))*A2y;

        B2x = ((t3-t)/(t3-t1))*A2x + ((t-t1)/(t3-t1))*A3x;
        B2y = ((t3-t)/(t3-t1))*A2y + ((t-t1)/(t3-t1))*A3y;

        Cx = ((t2-t)/(t2-t1))*B1x + ((t-t1)/(t2-t1))*B2x;
        Cy = ((t2-t)/(t2-t1))*B1y + ((t-t1)/(t2-t1))*B2y;

        x_cat(i) = Cx;
        y_cat(i) = Cy;
    end
end

end
```

B.3 Safety measure for crash avoidance method using Catmull-Rom curves (Variable horizon)

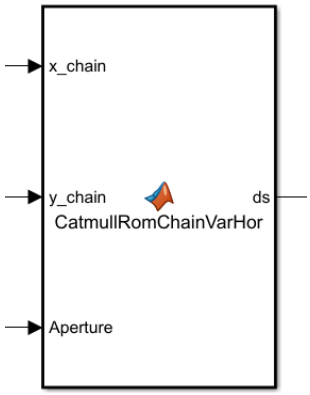

Figure B-3: Safety measure for crash avoidance method using Catmull-Rom curves (Variable horizon)

```
function ds  = CatmullRomChainVarHor(x chain, y chain, Aperture)
%Computes the intersection between the sensor scope and the fitted
%path (Catmull-Rom curves)
ds_set = 0;
dsx = 0;
dsy = 0;
Range = 1e6;

for i=1:length(x_chain)
    %%
    if ~ds_set
        if y_chain(i) >= 0
            ysens = x_chain(i)*tan(Aperture);
            if y_chain(i) >= ysens
                dsy = ysens;
                dsx = x_chain(i);
                ds_set = 1;
            else
                dsy = y_chain(i);
                dsx = Range;
            end
        elseif y_chain(i) < 0
            ysens = -x_chain(i)*tan(Aperture);
            if y_chain(i) <= ysens
                dsy = ysens;
                dsx = x_chain(i);
                ds_set = 1;
            else
                dsy = y_chain(i);
                dsx = Range;
            end
        end
    end
end
ds = sqrt(dsx^2+dsy^2);
```

B.4 Control point manager depending on the distance of the speed dependent moving point for curvature calculation

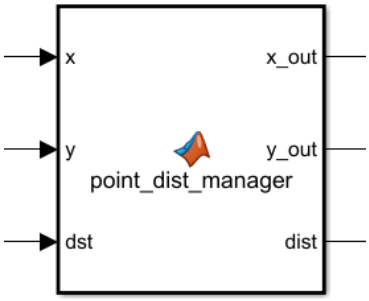

Figure B-4: Control point manager depending on the distance of the speed dependent moving point for curvature calculation

```
function [x_out, y_out, dist] = point_dist_manager(x, y, dst)
%Compares the distance required by the speed dependant point for
%curvature calculation, dst, with the control points. Then, decides
%which control points to use to calculate the curvature in the correct
%segment at the required distance dst
if dst<=x(2)
    x_out = x(1:4);
    y_out = y(1:4);
    dist = x(2);
elseif x(2) < dst && dst <= x(3)
    x_out = x(1:4);
    y_out = y(1:4);
    dist = dst;
elseif x(3) < dst && dst <= x(4)
    x_out = x(2:5);
    y_out = y(2:5);
    dist = dst;
elseif x(4) < dst && dst <= x(5)
    x_out = x(3:6);
    y_out = y(3:6);
    dist = dst;
else
    x_out = x(3:6);
    y_out = y(3:6);
    dist = x(5);
end
```

B.5 Curvature calculation at speed dependant distance

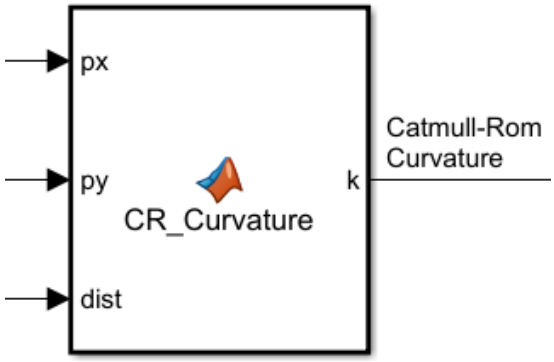

Figure B-5: Catmull-Rom curvature calculator

```
function k  =  CR_Curvature(px, py, dist)
    %Computes the second derivative of the fitted Catmull-Rom splines
    % to calculate the absolute value curvature at the desired dist
    alphaCR = 0.5;

    t0 = 0;
    t1 = (sqrt((px(2)-px(1))^2+(py(2)-py(1))^2))^alphaCR + t0;
    t2 = (sqrt((px(3)-px(2))^2+(py(3)-py(2))^2))^alphaCR + t1;
    t3 = (sqrt((px(4)-px(3))^2+(py(4)-py(3))^2))^alphaCR + t2;
    t = ((t2-t1)/(px(3)-px(2)))*(dist-px(2)) + t1;
    %%
    A1x = ((t1-t)/(t1-t0))*px(1) + ((t-t0)/(t1-t0))*px(2);
    A1y = ((t1-t)/(t1-t0))*py(1) + ((t-t0)/(t1-t0))*py(2);

    A2x = ((t2-t)/(t2-t1))*px(2) + ((t-t1)/(t2-t1))*px(3);
    A2y = ((t2-t)/(t2-t1))*py(2) + ((t-t1)/(t2-t1))*py(3);

    A3x = ((t3-t)/(t3-t2))*px(3) + ((t-t2)/(t3-t2))*px(4);
    A3y = ((t3-t)/(t3-t2))*py(3) + ((t-t2)/(t3-t2))*py(4);

    B1x = ((t2-t)/(t2-t0))*A1x + ((t-t0)/(t2-t0))*A2x;
    B1y = ((t2-t)/(t2-t0))*A1y + ((t-t0)/(t2-t0))*A2y;

    B2x = ((t3-t)/(t3-t1))*A2x + ((t-t1)/(t3-t1))*A3x;
    B2y = ((t3-t)/(t3-t1))*A2y + ((t-t1)/(t3-t1))*A3y;

    Cx = ((t2-t)/(t2-t1))*B1x + ((t-t1)/(t2-t1))*B2x;
    Cy = ((t2-t)/(t2-t1))*B1y + ((t-t1)/(t2-t1))*B2y;


    %%
    A1xp = (1/(t1-t0))*(px(2)-px(1));
    A1yp = (1/(t1-t0))*(py(2)-py(1));

    A2xp = (1/(t2-t1))*(px(3)-px(2));
    A2yp = (1/(t2-t1))*(py(3)-py(2));

    A3xp = (1/(t3-t2))*(px(4)-px(3));
```

```
    A3yp = (1/(t3-t2))*(py(4)-py(3));

    B1xp = (1/(t2-t0))*(A2x-A1x+(t2-t)*A1xp+(t-t0)*A2xp);
    B1yp = (1/(t2-t0))*(A2y-A1y+(t2-t)*A1yp+(t-t0)*A2yp);

    B2xp = (1/(t3-t1))*(A3x-A2x+(t3-t)*A2xp+(t-t1)*A3xp);
    B2yp = (1/(t3-t1))*(A3y-A2y+(t3-t)*A2yp+(t-t1)*A3yp);

    Cxp = (1/(t2-t1))*(B2x-B1x+(t2-t)*B1xp+(t-t1)*B2xp);
    Cyp = (1/(t2-t1))*(B2y-B1y+(t2-t)*B1yp+(t-t1)*B2yp);

    %%
    B1xpp = (2/(t2-t0))*(A2xp - A1xp);
    B1ypp = (2/(t2-t0))*(A2yp - A1yp);

    B2xpp = (2/(t3-t1))*(A3xp - A2xp);
    B2ypp = (2/(t3-t1))*(A3yp - A2yp);

    Cxpp = (1/(t2-t1))*(2*B2xp-2*B1xp+B1xpp*(t2-t)+B2xpp*(t-t1));
    Cypp = (1/(t2-t1))*(2*B2yp-2*B1yp+B1ypp*(t2-t)+B2ypp*(t-t1));

    k = abs(((Cxpp*Cyp-Cypp*Cxp)/((Cxp^2 + Cyp^2)^(3/2))));

    %% Nan manager
    if k~=k
        k = 0;
    end

end
```

B.6 Time managed crash avoidance safety feature

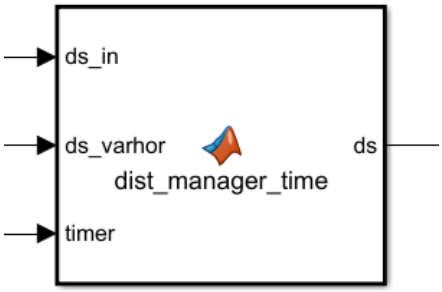

Figure B-6: Time manager for crash avoidance safety feature

```
function ds = dist_manager_time(ds_in, ds_varhor, timer)
%Outputs the intersection point distance if a vehicle dissappeared the
% sensor scope in less than 20s
if timer > 0 && timer <= 20
   ds = ds_varhor;
else
    ds = ds_in;
end
```

B.7 Acceleration manager depending on the relative velocity beyond the speed limit

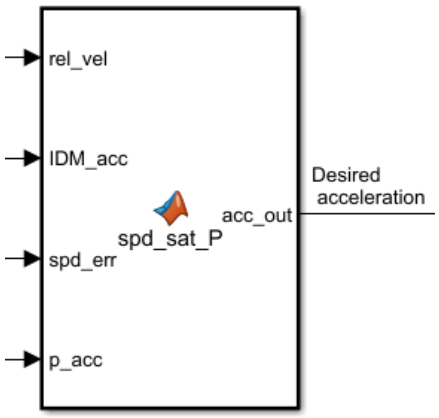

Figure B-7: Acceleration manager depending on the relative velocity beyond the speed limit

```
function acc_out = spd_sat_P(rel_vel, IDM_acc, spd_err, p_acc)
%Switches to a P controller acceleration that returns to the speed
%limit if the relatie velocity is positive beyond the speed limit
if spd_err <= 0 && rel_vel > 0
    acc_out = min(IDM_acc, p_acc);
else
    acc_out = IDM_acc;
end
end
```