Human-Driving Highway Overtake and Its Perceived Comfort: Correlational Study Using Data Fusion

Human-Driving Highway Overtake and Its Perceived Comfort: Correlational Study Using Data Fusion

Massimiliana Carello, Alessandro Ferraris, Henrique de Carvalho Pinheiro, and Diego Cruz Stanke Politecnico di Torino

Giovanni Gabiati, Isabella Camuffo, and Massimo Grillo Fiat Research Center

Abstract

As an era of autonomous driving approaches, it is necessary to translate handling comfort – currently a responsibility of human drivers – to a vehicle imbedded algorithm. Therefore, it is imperative to understand the relationship between perceived driving comfort and human driving behaviour. This paper develops a methodology able to generate the information necessary to study how this relationship is expressed in highway overtakes. To achieve this goal, the approach revolved around the implementation of sensor *Data Fusion*, by processing data from CAN, camera and LIDAR from experimental tests. A myriad of variables was available, requiring individuating the key-information and parameters for recognition, classification and understanding of the manoeuvres. The paper presents the methodology and the role each sensor plays, by expanding on three main steps: Data segregation and parameter selection; Manoeuvre detection and processing; Manoeuvre classification and database generation. It also describes the testing setup, and posterior statistical analysis. To perform all the steps MATLAB was chosen, serving as an all-in-one environment equipped with the necessary toolboxes and libraries to perform filtering, camera perception, operate on matrixes, and database generation. The resultant algorithms can extract manoeuvres, identify their subsegments (e.g. cut-out, cut-in) and isolate their contribution on the vehicle dynamics and comfort, such as supplemental lateral acceleration. Furthermore, they allow the comparison of different manoeuvres, by grouping them into scenarios conceived through an altered decision tree, in which the selection criteria assimilated a human driver's decision-making process. This methodology proved effective on extracting over 300 manoeuvres from 14 experiments, calculating all relative parameters, classifying them into statistically generated scenarios and originating a database useful for statistical analysis, machine learning and manoeuvre generation. In future development, increasing the amount of experiments and diversifying the vehicle types can create a more complete database and a more robust analysis shall be achieved.

Introduction

It is indisputable the relevance of Autonomous Driving and overall Advanced Driving Assistance Systems (ADAS) in the context of recent academic research. Some of these solutions also start to be seen in practical applications for high-end vehicles or prototypes aiming to proof the concept and show the state of the art of such technologies. Alongside with electrification and forms of alternative propulsion [1-7], lightweight and advanced materials [8-9], connectivity and sharing [10-19], the concept of self-driving vehicles is considered as one of the most capable to profoundly change mobility as known today. Evidently the technical challenges to be overcome are not simple nor few, but aside from the pure engineering issues, the testing and regulating of vehicles equipped with Level 4 or Level 5 automation (as classified by SAE)) pose another demanding task to carmakers, governments and agencies. In that realm, most of the effort seen so far (quite reasonably) concentrates in the safety of autonomous driving.

The work presented in this paper is a continuous part of [20] and try to tackle the problem of objectively evaluating the comfort associated with driving behaviour, factor today attributed to the human driver whose burden will shift for the vehicle and its internal algorithm once the self-driving capability fully deploys. In [20]) the authors promoted a series of experimental tests to assess the comfort characteristics of *human driving behaviour* and succeeded reproducing this behaviour with different trajectory planning methodologies.

This paper uses the same dataset to further improve the data analysis by means of a more robust and complete methodology, adding the *Data Fusion* concepts to aggregate the information of cameras and LIDAR to the ones of GPS and the CAN Bus previously explored.

Hereafter, the experimental setup will be quickly reminded and the techniques of detection, classification and analysis of overtake manoeuvres in highway environment described. The results are considered valuable to the scientific community and automotive industry since the understanding of driving comfort shall be increasingly important in the next phases of autonomous driving development. The detailed description of the algorithms presented can be an important piece to build up a robust and insightful methodology to analyse manoeuvres and generate databases with realistic and relevant testing scenarios.

Experimental Setup

This section is aimed to describe the experimental setup used for the data acquisition. Just the most relevant characteristics of the procedure are, therefore, presented.

Test Location and Tested Parameters

The scenario selected for those tests was the highway between Torino and Mondovì (Figure 1), because of its long straights and because in this segment the traffic conditions are not so heavy, allowing the execution of several manoeuvres with certain freedom in choosing the trajectory. The sensorized vehicle was provided by FCA group, and the procedure was simple: each driver must drive according to his personal style focusing the attention only on the speed limits and the other Road Laws.

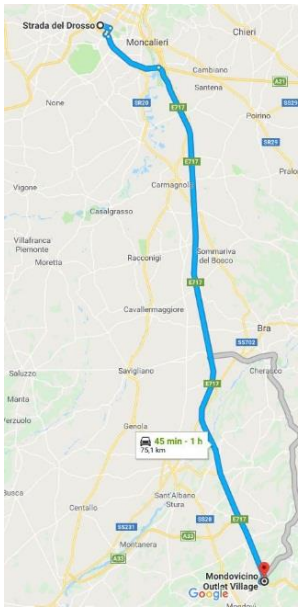

Figure 1. The highway route used for road testing

More than 700 overtake manoeuvres have been analysed, thanks to the contribution of seven different drivers, performing roundtrip runs on two different vehicles.

The cohort chosen to perform the experiment aims to present a high degree of heterogeneity, so seven drivers with different gender, profession, age and yearly mileage were chosen to verify if those parameters generate variation in the driving style and if those differences could be appreciated through the selected performance indicators.

Physical Setup

A simple scheme of the experimental setup is shown in Figure 2.

Most of the information had been collected from CAN network, through a CAN Network Logger. The vehicle dynamics data like accelerations, Yaw Rate, steering angle come from the sensors embedded on the vehicle, used by vehicle ECU to manage Passive Safety Systems like ABS and ESP. In Table 1 are reported the specification about the accuracy of sensors used.

CAN Bus and LIDAR information were passed through *CANanalyzer* in order to sync all of the different signals which arrived at the can as to have the same sampling rate of 100 Hz

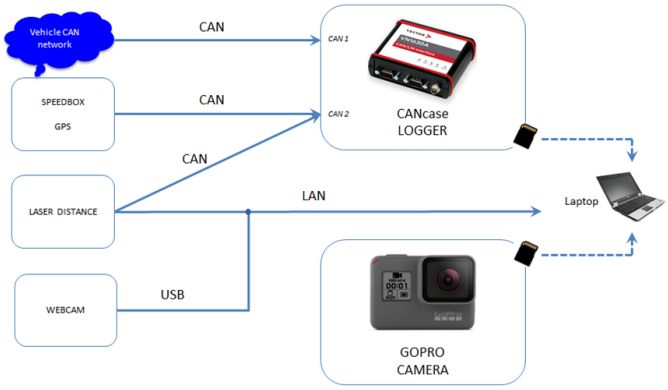

Figure 2. Experimental setup for the tests

Table 1. Sensors specifications

Yaw rate [Ψ]	Resolution	0.1		deg/s
	Measuring range	-100	+100	deg/s
	Accuracy	-5	+5	%
	Noise RMS	0.1	0.2	deg/s
Lateral acceleration [a_y]	Resolution	0.05		m/s^2
	Measuring range	-18	+18	m/s^2
	Accuracy	-5	+5	%
	Noise RMS	0.05	0.1	m/s^2
Steering angle [δ]	Resolution	0.1		deg
	Measuring range	-720	+720	deg
	Accuracy	1.4	2.8	deg
GPS	Absolute positioning Accuracy (CEP)	3		m
	Relative Accuracy	0.01		%
	speed accuracy	0.5		km/h

Camera calibration in MATLAB

Unless the system knows the optic properties of the camera used to record the video footage, it will not be able to measure distances correctly, something crucial for this study. Since it will not be able to cope with the distortions correctly nor will it know the position and orientation of the camera in the vehicle either and compensate for it.

A camera calibration must be performed and the MATLAB tool (*Camera Calibrator)* has been used, to do so, and it outputs the camera intrinsic, in other words, the focal properties (Figure 3).

It requires that a set of at least twenty images are taken of a checkerboard pattern at different angles and distances, which are then loaded into the program. Then it must be declared if the camera has a fisheye lens or not (so it can interpret the distortion better) and the length of the squares' edges. Once both those things are set, it proceeds with the calibration. What it essentially does, it identifies the vertices of the squares and see how far they are from the neighbouring ones and understands why they got closer or further away as a causality of camera optics. It ends up delivering a statistical result of the calibration and a file containing all the camera optics which will be implemented later.

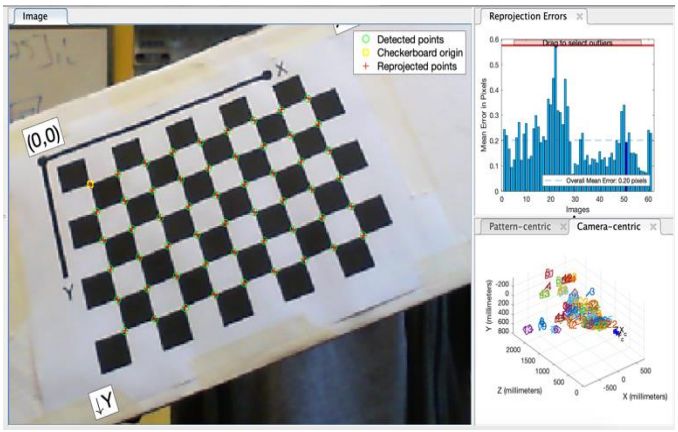

Figure 3. Camera Calibration Procedure.

Manoeuvre Detection

The data extracted from the experimental runs as it can be implied, holds the information of the entire run, so it was necessary to develop a proper methodology to find the individual manoeuvres and extract all the information relative to them in order to study them properly later. This procedure had to be design so it could process all the files automatically, because if not it would develop into a very tedious and difficult task, where very long arrays would have to be manually analysed and operated on.

Each of the experimental runs consisted of three information sources:

- CAN Bus data file in *.mat* MATLAB array structure format.
- Camera Video file in *.mp4* format. Plus a syncing file.
- LIDAR data contained in CAN Bus data file.

Data Pre-Selection and Parameter selection

Not all the data provided inside the three main source files (CAN Bus, Camera and LIDAR) will be used in the study and loading all of it becomes memory intensive, so a pre-selection of the data (Table 2) that will be used must be performed. This is done in the following set of steps, in which step three onwards are ran again for each experimental file:

1. *Pre-loading of data files.*
   The location of the source files of all the experiments are loaded, and the system now knows all the number of experimental runs it must analyse.

2. *Selection of Parameters and tolerances.*
   Many functions and algorithms have parameters that need to be tweaked in order to work correctly or in a desired way with the analysed files, so these are left at the beginning, making them easily reachable. Here things like data compression ratio, yaw angle tolerances, and interval range of analysis are defined, for example.

3. *Loading of experimental source data*
   Once a specific experimental file is being analysed, then its source files are loaded into the system.

4. *Extracting the relevant CAN Bus data*
   Data that can give an understanding of vehicle dynamics, position, event triggers are all extracted in order to help finding the manoeuvres, characterize them and study them after. A list of the variables chosen is shown in step 6.

5. *Generate time compression array reference*
   In order to reduce memory usage in the system and to speed up the analysis considerably, the data arrays of the CAN Bus (LIDAR information is not compressed here, but in a later stage) are reduced from the original sample rate of 100 samples per second, down to 20, so five times smaller arrays are obtained. This is done by getting the mean value of the corresponding section of data and setting that as the value for the compressed index.

6. *Variable generation*
   The data extracted is treated by generating a compressed version of it and is converted to the proper units to be of use to this study, like for example GPS values are passed from longitude and latitude to values in meters (referred to a world map). Table 2 displays the final variables selected.

Table 2. CAN Bus and LIDAR generated variables

Variable	**Units**
Time (various arrays)	hh:mm:ss, s, 1/20 s
GPS Latitude + Longitude	m
X, Y Trajectory	m
Longitudinal Acceleration	g (m/s^2)
Lateral Acceleration	g (m/s^2)
Speed	m/s
Steering Angle	Rad
Yaw Rate	rad/s
Right + Left Turn Light Activations	indexes + timetable
LIDAR Object Ids	number identifier
LIDAR Object positions X,Y	M
LIDAR Objects Height & Width	M

Possible Manoeuvre Candidate Detection

In order to perform an overtake (Figure 4), the driver must alert the other vehicles that he will do so, or at least that he intends to do so through activating the corresponding turn light. In this scenario, since the overtakes are performed along the left side, the left indicator is the corresponding one, and the right indicator will be activated when the driver decides that he plans on returning to the original lane.

This is especially useful when it is required to find the overtake inside the CAN data, since the state of both turn lights is registered in every step, as a 0 if it is off, and as a 1 if it has been activated. Activation flags (values = 1) can also include cases where the indicator might have been selected but cancelled almost immediately after. This problem is avoided by looking for activations inside the turn light activation arrays followed by at least twenty consecutive indexes containing ones, meaning that the indicator was active for at least one second. This is performed for both turn lights and a timetable has to be generated with pairs of left and right turn light activations. These pairs are composed by all the left activation followed (in time) by a right activation and not a left one.

Each row of this timetable then helps in designing a search area in which to search for possible manoeuvres. Only the time at which the turn lights were activated is known, but by defining a range of time sufficiently large after the right light has been activated can almost guarantee that the entirety of the overtake will be contained between this generous predicted end time and when the left turn light was first activated. The search areas for each time interval are generated as follows:

- Lower Search Bound:

$$Lower_{bound} = \mathrm{LTLA}_{\mathrm{time}} - 4 \text{ seconds} \quad (1)$$

- Upper Limit:

$$\Delta TimeTurnLightActivations = RTLA_{time} - LTLA_{time} \quad (2)$$

$$TimeToEnd = 1.7 * \Delta TimeTurnLightActivations \quad (3)$$

$$if\ TimeToEnd \geq 50s \rightarrow TimeToEnd = 50s \quad (4)$$

$$Upper_{bound} = \mathrm{RTLA}_{\mathrm{time}} + TimeToEnd \quad (5)$$

The reason that the search ranges are not larger, as to for sure guarantee finding the entirety of the overtake (if there is one, of course) is to reduce computational power and processing time. Various filtering, fitting and matrix operations are performed, and having longer arrays is detrimental to performance, and even more when there is video analysis as well. Second reason is that if too large, sections of other overtakes might be included in the analysis, which might confuse the system producing errors if incorrectly handled. The time range selected before the left turn light is in order to

have enough information to regressively fit curves or tendencies in future steps. The generous time range set after the right turn light will be shortened to a couple of seconds after the time occurrence of the right lane change once it is detected through visual information.

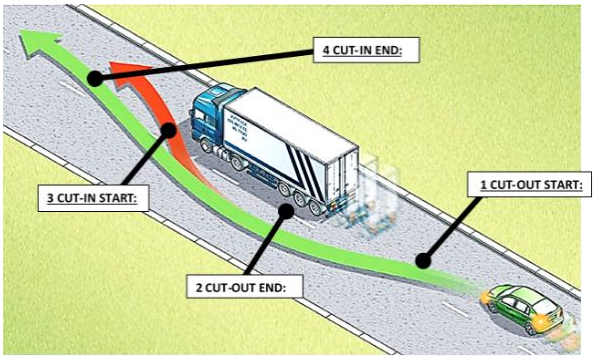

Figure 4. Overtake Manoeuvre main events.

Nevertheless, search ranges must be filtered if their length is not reasonable for this study. Cases where both turn lights were activated within less than five seconds or if the total search range accounts for 170 seconds, are discarded and tagged as not valid. In the end, the minimum requirement is that it includes all the elements displayed in the Figure 5.

Camera Lane-Tracking Analysis

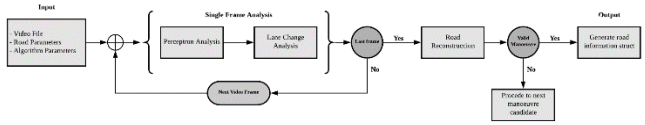

Figure 5. EGO vehicle trajectory extraction from Video File.

The exact set of steps that make up the frame analysis procedure, as well as the required inputs and generated outputs are describe in the procedure described below, but before it is important to clarify how the lanes are kept track of.

A table representing all the lane borders is created, so for example, for a two-lane highway, it would have three columns, plus one extra for the time of detection, where the vehicles starts with columns two and three initially representing the left and right lane borders. So that when a left lane change occurs, the data inputted is now shifted one column to the left, so the left lane border would be represented by column one and the right detection would be stored in column two.

1. *Perceptron analysis*
   MATLAB has a built-in script setup which allows to perform lane detection (Figure 6), but this must be configured properly to work with the experimental setup and scenario conditions. For example, detection sensitiveness, detection area, camera properties, lane dimension, etc. This procedure outputs the distance of the detected lane borders (left and/or right).

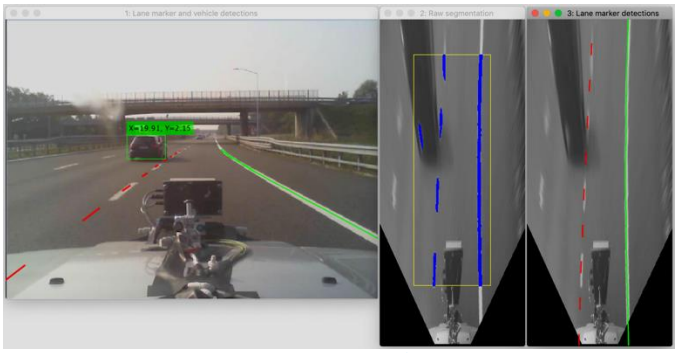

Figure 6. Perceptron analysis on experimental footage.

2. *Perceptron data filtering*
   Data incoming from the perceptron system is incredibly noisy, so every detection must pass through a series of checks in order to be accepted. For example, if both lane borders are detected, the distance between them must be inside a certain range (Lane Width ± a tolerance). For single lane border, depending on their location, the conditions to be met so it is accepted vary, like if it is in a lane change zone it has stricter criteria than if it was in the normal lane range.

3. *Detection correction and compensation of missing values*
When both lane borders are detected, these are recalculated so to be exactly a lane width apart, through the following procedure:

$$Lane\ Centre = \frac{LD+RD}{2} \quad (6)$$

$$LD\ =\ Lane\ Centre + 0.5 * Nominal\ Lane\ Width \quad (7)$$

$$RD\ =\ Lane\ Centre - 0.5 * Nominal\ Lane\ Width \quad (8)$$

If only one lane border is detected, depending on the position on the lane border it is (there are a series of variable cases) the other lane border's distance is calculated as:

$$LD = RD + Nominal\ Lane\ Width,\ if\ RD\ is\ valid \quad (9)$$

$$RD = LD - Nominal\ Lane\ Width,\ if\ LD\ is\ valid \quad (10)$$

To be able to recreate the entirety of the road, also the distance of other lane boundaries which might exist in the scene are calculated and will be eliminated later if not relevant.

$$LBD_i = (-i + CurrentLB_i) * Lane\ Width_{nom} + LD \quad (11)$$

4. *Lane change analysis*
Once the detections have been filtered and corrected, they are analysed in order to verify if a lane change occurred or not. It is a two-step verification process, where first the detection position is checked and compared with the previous value, and if it finds the correct conditions for a lane change to have occurred, it passes to the verification step, where the trajectory tendency is analysed to corroborate that the vehicle was in fact heading towards a lane change.

To understand better the first phase, which is a possible lane change identification. Figure 7 describes the identification of a left lane change (the right lane change case is a mirror of this) and then the brief explanation that follows.

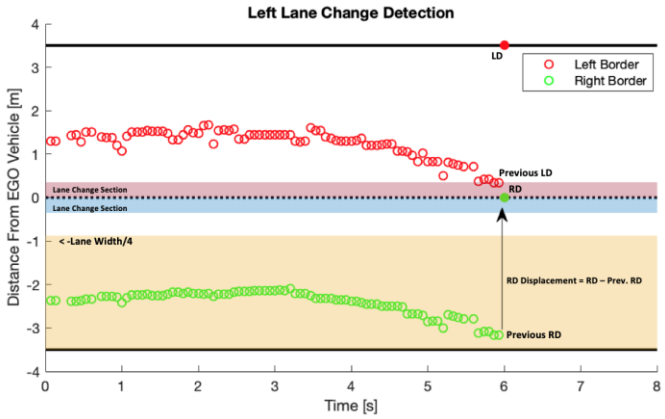

Figure 7. Identification of possible left lane change.

Using the left lane change case as an example (the right lane change is analogous). The system is constantly checking the position of the detections, and if it detects a right border detection in the *lane change section*, a flag is raised, and it will to check the following set of conditions:

- No lane change occurred within the previous second.
- Previous left border detection is in its respective lane change section.
- Previous right border detection is in the yellow range (detection distances < -0.25 • (Nominal Lane Width)).

There is another possibility (in order to cope with conditions where the perceptron outputs less points), where if the right border detection is outside the *lane change section*, but the following conditions are met, a possible lane change is considered:

- No lane change has occurred within the previous second.

- Previous left border detection is in its respective lane change section.
- RD displacement ≥ 0.5 • (Nominal Lane Width)).

The second phase is to comprehend if the vehicle trajectory tendency displayed the corresponding lane change intention. The approach is to analyse up to the last forty detections corresponding to the lane border the ego vehicle approached. A second order polynomial curve is fitted to the data, and the mean values for the first and final thirds are computed, and compared as follows:

- For a left lane change analysis:

$$\overline{(1st\ third)} > \overline{(3rd\ third)} \rightarrow Lane\ Change \quad (12)$$

- For a right lane change analysis:

$$\overline{(1st\ third)} > \overline{(3rd\ third)} \rightarrow Lane\ Change \quad (13)$$

5. *Lane change update (only if a lane change occurred)*
Since the study is focused on analysing overtaking manoeuvres, it is necessary to check if the vehicle is performing the corresponding manoeuvre. This means, that it performs first a left lane change and then a right one, any other order will automatically disqualify the analysed candidate section of the data, and the algorithm will proceed with the next one.

   Next it must update the column indexes related to the lane borders it is filling into the table representing the lane border distances throughout the analysis described previously. It also stores the time at which the event happened, so it can be used in future operations.

   One important operation done at this point, is that if it detects a valid right lane change, it updates the end time of the analysed section so that it occurs with the same delay it took the vehicle to perform the left lane change or until it encounters another lane change. This way the tentative value set at the beginning is corrected, and only the section of interest will be studied.

6. *Last Frame Check*
The system checks if the defined time interval for the video analysis has been completed or not. If not, it re-runs the procedure from step one for the following frame. If it has finished, it continues to the next process.

Once all the valid detection points of the trajectory have been gathered, it is then necessary to reanalyse them, smooth them out and approximate missing values. The steps are as followed:

1. *Define a smoothing scale proportional to the video length*.
The smoothening process uses a second-degree polynomial least squares model local regression model, which requires a local vicinity (a percentage of the data) to perform the regression of each point. For better results, the vicinity size is dictated by an equation, where it assigns smaller size vicinities to faster trajectories as these contain less data and over smoothening may occur with larger vicinities.

2. *Smooth and denoise the data to generate the trajectory.*
The table containing the lane limit distances detected is analysed column by column, and the values are smoothed using the technique mentioned in step 1, where the time reference used to smooth is the time-of-detection column.

3. *Change reference system of the trajectory*.
Up to now the values represent the distance of the lanes with respect to the vehicle, but it is required to obtain the position of EGO vehicle with respect to the centreline. This is done by adding a negative sign to the values in column 2 and storing this into a new array. The lane border array positions relative to the centreline are obtain by subtracting the values of the column referring to the centreline (column 2 of the table) to their respective columns.
The final result can be seen in Figure 8.

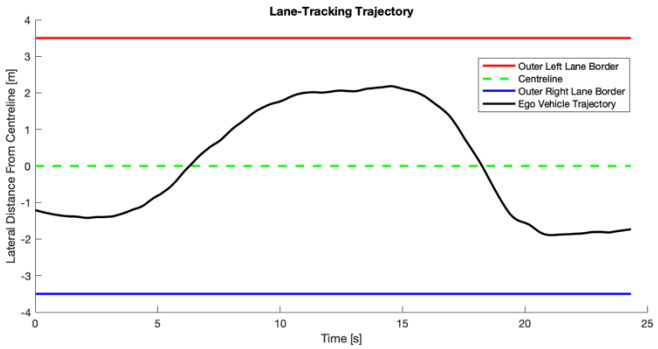

Figure 8. Resulting trajectory from lane-tracking.

4. *Remap the trajectory to match the CAN sampling rate.*
   The frame rate of the video is not the same as that of the CAN, so all the data that will be studied alongside the CAN data has to be mapped from one time-space to the other. The mapping is done by means of a one-dimensional interpolating function in which the time array the values of the acquisitions, as well as, the time array these will be mapped on, serve as inputs, and the output will be the mapped values.

5. *Final manoeuvre validity check.*
   Even if a valid manoeuvre shape (behaviour) has been found, it still requires a final check in order to be approved. Both turn signals have to happen before their respective lane changes, if not, the manoeuvre is not considered, since it would not represent a correctly performed manoeuvre. Nevertheless, if it is a persistent error throughout the experimental file, it may mean that the tentative value of time lag of the video file with respect to the CAN Bus acquisition time might still be too far away from the real value.

Once the camera-based trajectory and all its data and parameters of interest have been identified (as shown in the Table 3) and correctly processed, it is then possible to study and process the CAN Bus and LIDAR data.

Table 3. CAN Bus and LIDAR variables generated

Output	**Description**
Real time interval of interest	Actual start and end time of the trajectory section of interest.
Vehicle trajectory (lateral position evolution in time)	Lateral distance of the ego vehicle to the road centreline through time.
Lane boundaries	Distance of all the lane boundaries with respect to the vehicle (and centreline).
Lane change time indexes	Time index of both the left lane change and right lane change.

CAN Bus Data Filtering and Cleaning

Data acquired through the vehicle sensors and then stored in the CAN Bus is affected by noise due to various different reasons, and this might become an issue when performing data analysis, since outliers are likely to alter results, and, consequently, the interpretation of those results. Thus, a data treatment must be done to filter and smooth out the many signals present in the CAN file which are intended to be used in the study.

The reason this is not performed in a prior stage, where all time history of the CAN data would be treated at once, is that by only treating the interval of information being analysed. It is less computationally intensive and overall better results are obtained since filter setup used can better treat data when it is relatively short and somehow correlated. This basically means that CAN data relative to a specific manoeuvre is only loaded and filtered once the interval of interest is known, in order not to load unnecessary data which at the end is not used and will only use resources.

The filter procedure is:

1. Missing or erroneous data entries of the GPS trajectory due to jumps or misreads are replaced by values estimated through an autoregressive function.
2. Design a Butterworth low-pass filter.
   a. A 4^{th} order low-pass filter is used.
   b. Normalized Cut-off frequency of 4 hertz selected for the 20 Hz sampling rate of the data.
3. Filter and denoise the signals.
   a. Generate a filtered copy of each of the CAN Bus variables of interest.
   b. Denoise each of the filtered signals to reduce small perturbances which the low pass filter is not able to eliminate.
4. Smooth out the resulting filtered GPS trajectory. Since it changes at a lower frequency than the rest, it requires a final tuning.
   a. Apply a second-degree polynomial least squares model local regression model to smooth the data, where the outliers are ignored.
   b. Apply a second smoothening operation, but with a second-degree polynomial least squares model local regression model that considers outliers.

CAN Bus – Camera Data Syncing and Data Fusion

Data acquisition on the CAN bus network and on the camera were not colligated, meaning these were not synchronized, and it was necessary to find a way to link and sync both sets of data. The approach selected consisted on defining a time range inside the can data where it was highly probable to find the point where the camera began recording and select a value inside this range as a first tentative lag time of the video with respect to the CAN Bus data.

To find the exact time lag or at least a value very close to it, became necessary to find two similar data structures or values which had a strong relation in both sets of data to compare and find an offset. For this specific case, since the GPS trajectory was included in the CAN bus dataset and the car trajectory could be extrapolated from the camera data and treated to be comparable with the former, it was a good option, but it was not that straightforward. Since the shape of the trajectory in both cases depends on the portion of the data being observed (the actual time interval), the offset calculated should be checked in a following iteration to certify that it was correct, and not a misread due to lack of information.

The iterative process is defined by the following sequence:

1. *Camera-based trajectory parameter calculation*
   Using the vehicle speed information from the can it is possible to obtain the distance travelled by the vehicle alongside its trajectory. Then for every time-step, calculating the distance travelled and lateral displacement per time-step, and applying a simple trigonometric approach both the yaw angle and longitudinal distance travelled (parallel to the road) are obtained.

- Yaw angle:

$$\beta_t = \arcsin\left(\frac{\Delta\text{Distance}_{\text{lateral}}}{\Delta\text{Distance}_{\text{total}}}\right) \quad (14)$$

- Longitudinal Distance Travelled:

$$\Delta\text{Distance}_{\text{longitudinal}} = \Delta\text{Distance}_{\text{total}} \cdot \cos\beta_t \quad (15)$$

$$\text{Cumulative Longitudinal Distance} = \sum_0^t \Delta\text{Dist}_{\text{long}_t} \quad (16)$$

2. *Identification of overtake limits in camera-based trajectory*
   By using the yaw angles and a set of tolerances it is possible to implement a search algorithm to identify the overtaking limits. The overtaking start is defined as the index following the index in which the yaw angle is lower or equal to the set tolerance angle for the last time between the left indicator light activation and the left lane change index. If no index is found, this value is set equal to the activation of the left turn light index. The overtaking end is defined as the index following the index in which the yaw angle is higher or equal to the set tolerance angle for the first time after the right lane change index. If no index is found, this value is set equal to the end of the manoeuvre.

3. *Identification of overtake limits and posterior reorientation and realignment of GPS Trajectory, in order to be able to compare both trajectories.*
   Overtake limits are found through an iterative algorithm that is able to search for local minima representing both overtaking limits on a slanted trajectory.

   Before this, it defines that the starting points for this iterative search are the same indexes representing the overtaking limits found in the previous step, which in case a lag is present, will not match with this trajectory's limits. An initial search interval for each point is also defined, so that the local minima is applied only in that section of the trajectory. This search area becomes progressively smaller as it tends toward the real limits, by applying mouldable learning rate which adapts to the search's evolution.

   a. Performs a local minima search (Figure 9) in each search area for new overtaking limits.

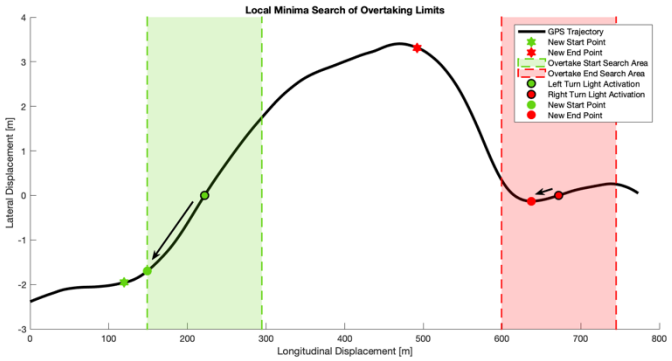

Figure 9. Local Minima Search Bounds.

   b. If new points are found, a reorientation and repositioning of the trajectory (Figure 10) is performed in order to have both limits lying on the line that passes through y = 0.

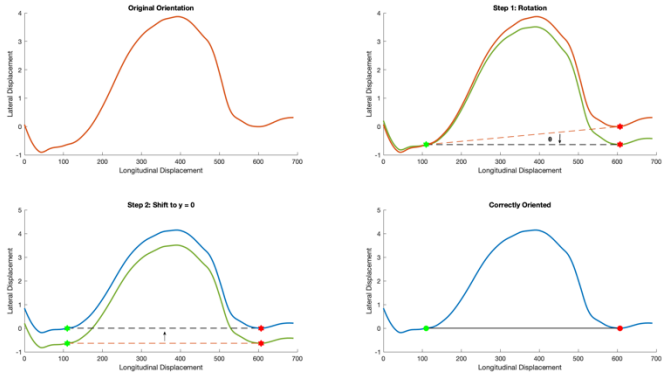

Figure 10. Trajectory reorientation and repositioning around candidate overtaking limit points.

c. After analysing the overtaking limits and comparing them to the previous values, there are three possibilities:

- The rate of change of the points is higher than a defined value, so the search boxes are decreased by the *normal* learning rate, and the search algorithm is ran again on the reoriented curve.
- The system analyses the rate of change of both overtaking limits, and if it detects that it starts to become stagnated, so that it is very near to the result, it increases the harshness of the search algorithm and does a limited number of extra runs while this assumption is true. If so, after the runs have been completed, it defines that the limits have been found.
- If the change is zero, it knows that it has found the overtaking limits.

The reorientation and shift performed will also allow to easily compare both trajectories later once the camera-base trajectory is also reoriented.

4. *Road profile extraction from GPS data.*
It is important to remember that the trajectory obtained from the camera is relative to the road the vehicle was travelling on, and the road is considered straight. The GPS trajectory is influenced by the road profile, so a technique was developed to extract it. The system analyses the portions of the trajectory before and after the overtake and fits 4th degree polynomial curve, which will represent the estimated road profile for a case where it is assumed that the vehicle departed and arrived at the same lateral position in the lane. Luckily, this is later compensated by overlaying the actual lateral position of the vehicle in the next step.

5. *Road profile overlay on camera-based trajectory.*
The lateral position described in the camera-base trajectory (Figure 11) is overlaid on the road profile extracted previously, and then this new trajectory is reoriented as to have both overtaking limits laying on the line parallel to the x axis that crosses y = 0. This allows it to be compared to the GPS trajectory (Figure 12).

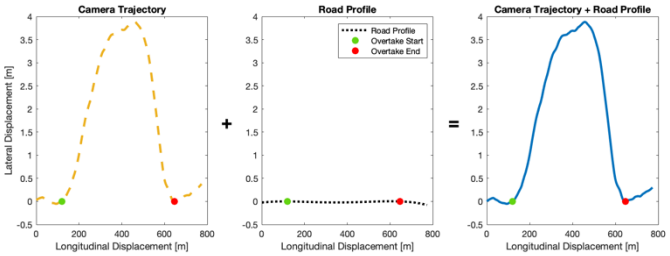

Figure 11. Road Profile Overlay on Camera Trajectory.

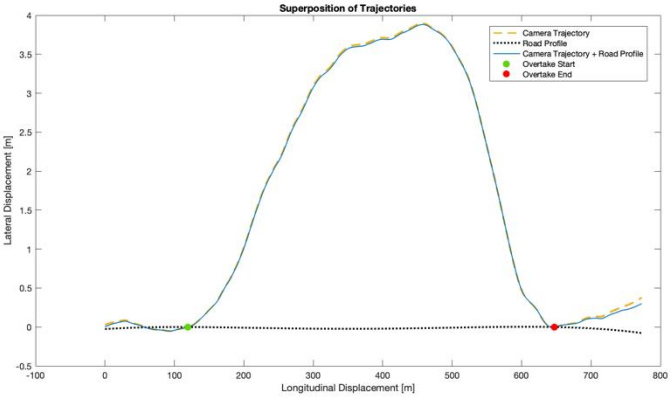

Figure 12. Superposition of the original camera trajectory, road profile and the new camera trajectory with road profile information.

6. *Time lag calculation*
   A cross-correlation analysis is performed between the two signals, to find by how much one of the signals must be shifted in order to get the highest correlation between the two sets of data (Figure 13). This shift is then understood as the lag between the shifted signal and the other one.

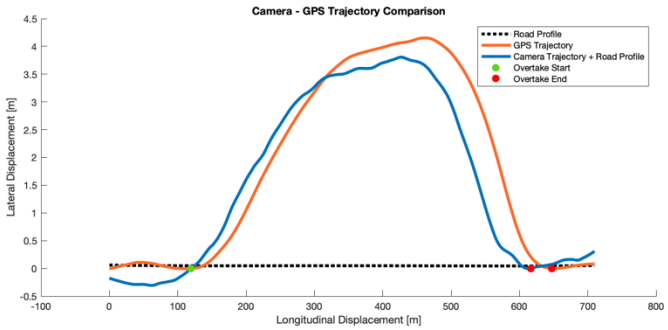

Figure 13. Trajectory Similarity Comparison

7. *Time lag compensation and verification.*
   The system analyses the calculated lag through the case check shown in Table 4 and in Figure 14.

Table 4. Time Lag checking.

Case	Action Performed
Time Lag > 0.1 s	Compensate lag and re-run the video analysis and all the steps up to this point.
Time Lag <= 0.1 s	Lag is negligible, thus continue with next steps.

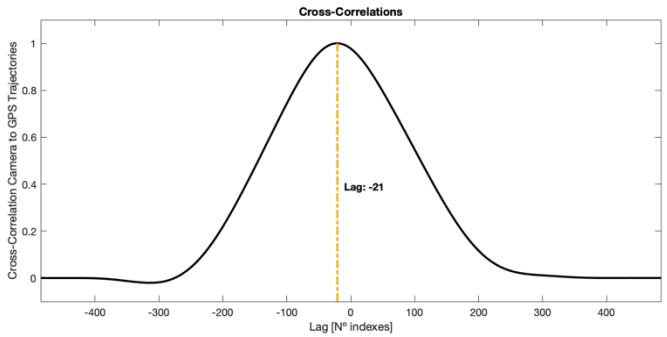

Figure 14. Cross-Correlation Lag Analysis.

The addition of the road curvature is also necessary to correctly position lidar objects in the scene, since road curvatures must be taken into account when positioning an object, a considerable distance ahead, as to know if it is in one lane or the other, or neither (Figure 15).

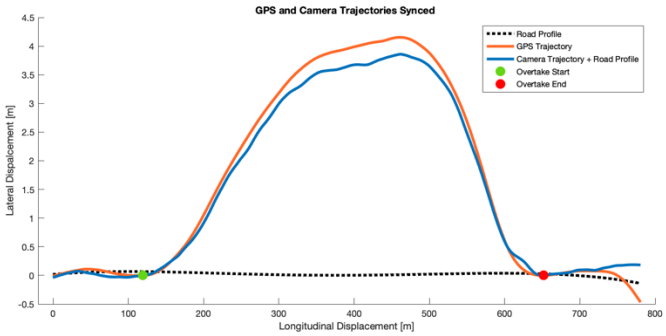

Figure 15. GPS and Camera Trajectories Synced.

Identification of Manoeuvre Subsections

It is also important to decompose each of the overtakes into its different sections, to study and understand them and see how these might affect the overall trajectory as well. Not only that, but this allows to group trajectories with very dissimilar overall shape together, since these might share a similar subsection.

Therefore, the subsection trajectory has been designed (Figure 16 and Table 5).

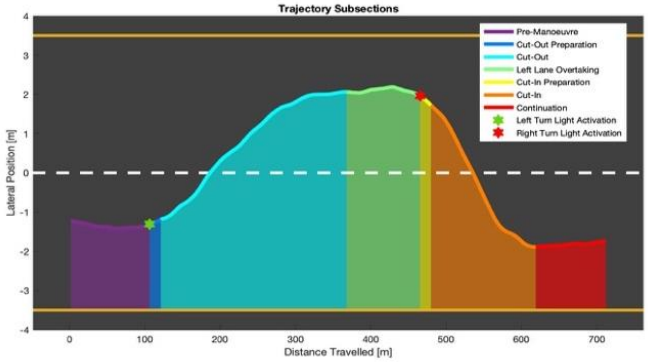

Figure 16. Trajectory Subsections

Table 5. Trajectory Subsections Explained

Subsection	**Description**	**Limits**
Pre-Manoeuvre	Period prior to the activation of the left turn signal.	L_{lim}: Start U_{lim}: LTLA
Overtake Preparation	Describes the period between the activation of the left indicator light and start of the overtake itself. The driving is preparing and waiting for the perfect moment to start the actual manoeuvre.	L_{lim}: LTLA U_{lim}: OV_{start}
Cut-Out	Transition from the right lane to the left lane. Lasts until the vehicle is stabilized and travelling parallel to the left lane.	L_{lim}: OV_{start} U_{lim}: CO_{end}
Left Lane Overtaking	Time frame between the cut-out end and the activation of the right indicator light, where the vehicle performs the surpassing of the vehicle in the original lane, or at least part of the surpassing.	L_{lim}: Start U_{lim}: RTLA
Cut-In Preparation	After triggering the right lane turn light, the driver is preparing for the right moment to commence the return to the original lane.	L_{lim}: RTLA U_{lim}: CI_{start}
Cut-In	Transition from the left lane to the original lane and last portion of the overtake itself. It lasts	L_{lim}: CI_{start}

	until the vehicle is once again heading parallel to the objective road.	U_{lim}: OV_{end}
Continuation	The vehicle trajectory development posterior to the overtake.	L_{lim}: OV_{end} U_{lim}: End

In order to define each of these limits it is necessary to study the yaw angles of the car with respect to the road centreline, as well as, identifying important indexes/events of the overtake. As explained when describing the search algorithm which uses yaw angles to find the overtake limits on the camera data, an almost identical approach is used here to find where the cut-out manoeuvre finishes and where the cut-in manoeuvre starts. For the former, the system searches for the index previous to the first index after the left lane change has occurred where the yaw angle is lower or equal to zero, meaning that the vehicle as stabilized. The latter is represented as the index following the last index after the right turn light was triggered, in which the yaw angle is equal or greater to zero, as to represent where the car starts to return back towards the original lane.

Once the left turn light activation, cut-out start, cut-out end, right turn light activation, cut-in end, and trajectory array end point indexes have been identified, it is possible to locate each of the subsections.

Overtaken Vehicle Data Extraction from Lidar and Fusion with Trajectory Data

Although possible to detect other vehicles with a mono-camera like in the setup used, it is not very precise in correctly measuring the distances between them and the ego vehicle and is sensible to lighting conditions and different vehicle types unless well trained and specifically designed for the task. The addition of a LIDAR (Light Detection and Ranging sensor) allows for the precise detection of other objects in the vehicle's surroundings. For this study, the main object of interest to keep track of, were the vehicles been overtaken, and this sensor allowed to detect their longitudinal and lateral position with respect to the vehicle, and thus, their velocity. Unfortunately, due to the use of a front facing radar, only the objects ahead of the vehicle could be detected, but if a 360º LIDAR is used. It is also potentially possible to keep track of vehicles travelling behind or next to the ego car, and study how another incoming vehicle might cause a variation in the behaviour of the driver.

The LIDAR's measurements were passed through the CAN Bus to keep them in sync with it, but this brought a series of problems due to the bandwidth limitations of this system. The main issue was that even though the LIDAR setup used is able to keep track of 256 objects at once, it could send to the CAN the information relative to only one of this per acquisition and was limited to the CAN's sampling rate as well, meaning that a lot of information was lost. Another important problem regarding the LIDAR's internal tracking algorithm of objects was that after losing track of an object and regained it back after a brief time, it was not able to interpret if the object was the same object it had previously detected moments ago, so it reassigned it another object id. This meant that one single vehicle could be represented by a myriad of object identifiers throughout its detection history.

To solve these issues the following procedure was identified and followed:

1. *Calculate the absolute positions of the detected points*
   The LIDAR setup used is able to place the objects in a reference frame relative to it, so to know their position in the road (Figure 17). So, in the absolute reference frame in which the ego vehicle trajectory is the one which combines the camera trajectory with the road profile, it became necessary to perform the following operations for each detected object:

   First to rewrite the coordinates of the LIDAR point from the LIDAR reference system to that of the ego vehicle.
   Then to calculate the distance between the detection and the origin of the vehicle reference system. After that, calculate the angle (α) between the line that connects the detected object's positions with the ego vehicle reference system origin and the longitudinal axis of this reference system; and add this angle to the respective yaw angle (β) (for the timestep of the detection) of the ego vehicle to obtain the absolute angle (θ) between the line that connects the detection and the ego vehicle with a line running parallel to the x axis that crosses the ego vehicle centre.

   $$\theta_{detection} = \beta_t + \alpha_{detection} \quad (17)$$

   Finally calculate the absolute position of the detection with the two following equations:

   $$x_{position} = Ego_x + dist_{detection} \cdot \cos\theta_{detection} \quad (18)$$

   $$y_{position} = Ego_y + dist_{detection} \cdot \sin\theta_{detection} \quad (19)$$

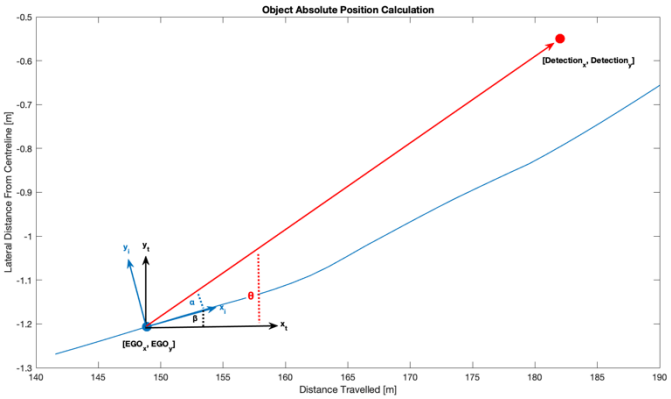

Figure 17. Absolute position referencing of detection objects.

2. *Define a search area of interest.*

   Since the overtaken vehicle travels through the right lane it is unnecessary to analyse objects present outside this lane. It was determined to only consider detections which are at up to 150 m away from the vehicle, as to consider only a reasonable visible range for the driver.

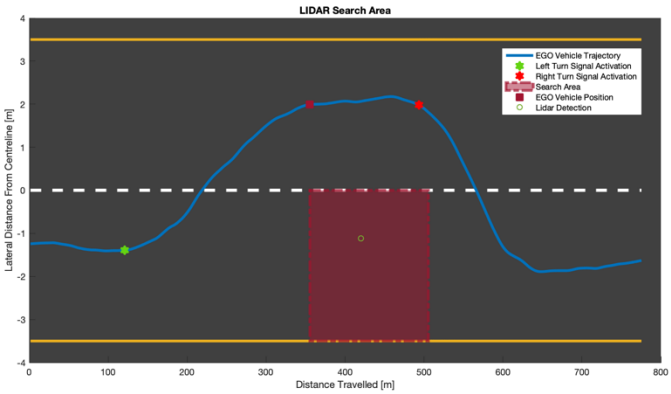

Figure 18. LIDAR search area for a specific instance of the EGO vehicle's trajectory.

   Therefore, a search box (which is not a box, but rather, follows the contour of right lane's borders) of length 150 m immediately in front of the vehicle is generated for each individual timestep of the analysed interval (Figure 18).

3. *Filter out objects outside area of interest.*
   The LIDAR absolute positions are analysed one timestep at a time and it is verified if they are inside or not of the respective search box. If so, all the information relative to the detected point is stored into a set of arrays which only include valid detections, to only take these into account for the posteriority of the analysis.

4. *Calculate all the detection positions as relative to the road centreline.*
   Once all the valid detections are known, it becomes significantly easier to perform the posterior steps if the lateral positions of the objects are all relative to that of the centreline. The process basically involves finding the normal distance between the detection and the centreline at the same absolute longitudinal position. For each detection (Figure 19) this relative lateral position to the centreline is calculated and stored inside a new relative lateral position array, like that of the ego vehicle.

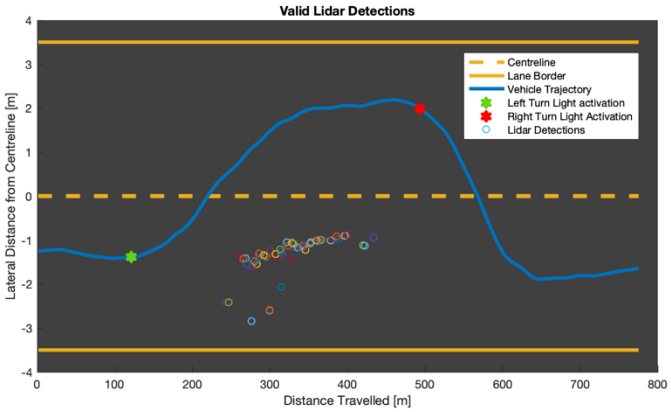

Figure 19. Valid LIDAR detections.

5. *Relate detection points and define vehicle objects.*
The system still believes that there are many different objects in the scene, because it has detected many different object identifiers, when it could be that there are only a couple or even just one vehicle present. It is then crucial to then analyse all the detection points in a specified order to check if some type of tendency exists which can correlate points as part of the same object.

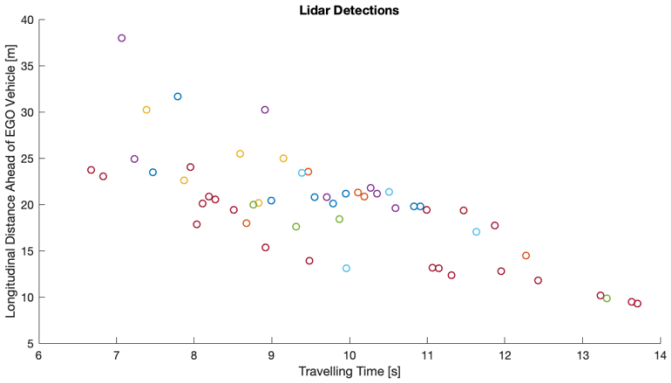

Figure 20. LIDAR detection distances to the EGO vehicle in time.

The approach selected was to analyse the longitudinal distance between the detected points and the vehicle as it travelled, thus allowing the system to *see* if the points tended to get closer or further away from the vehicle as time progress. This becomes a clearer by looking at Figure 20. It can be interpreted that there is a group of points, in which if a tendency line is drawn through them, this line could represent the longitudinal distance evolution of another vehicle with respect to the ego vehicle.

An algorithm was developed based on the same principle as a sliding window approach but tweaked in order to consider vehicle travelling speed constraints. It is explained through the following scheme:

a. Generate search window which origin lies alongside the same time position of the last known point of the identified object and longitudinal distance corresponds to the tendency of the points. (Figure 21)

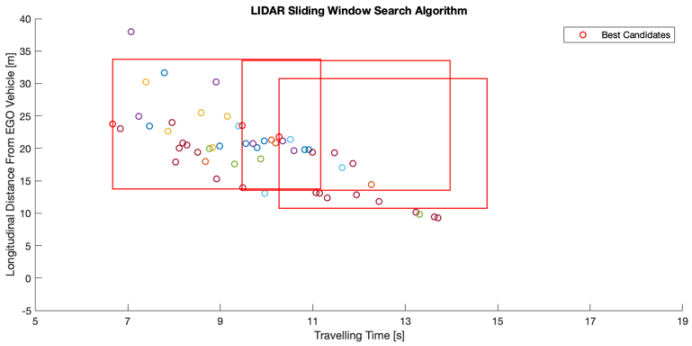

Figure 21. LIDAR sliding window search algorithm.

b. Conclusion of the current window search where possibly all principal tendency-derived object detections are found (Figure 22).

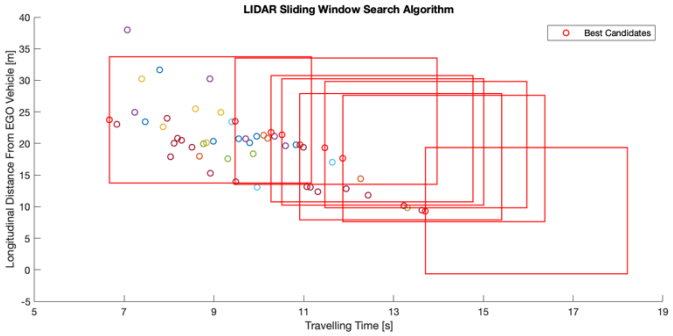

Figure 22. Completed LIDAR sliding window search algorithm.

c. Generation of detection band based on tendency of principle detection points and inclusion of all detections found within the band as detection related to the same object. (Figure 23)

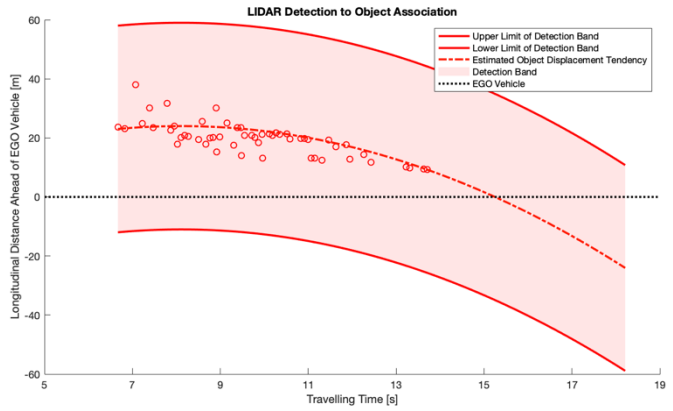

Figure 23. LIDAR object identification band.

At the end an array structure contains the final vehicle objects and all the points (and their respective information) belonging to them.

6. *Correct vehicle boxes and trajectory position.*
The LIDAR can compute the width and length of the detected objects as long as it has a clear visual of them, which of course it is not always the case. Usually when the detected object is right in front of the ego vehicle, the sensor does a really good information in estimating its width, but not its length. Contrarily, when the sensor has a side view of the vehicle, it can measure the length very well, but not the width (Figure 24). This is a problem, especially since the objects positions are referenced to the bounding box the system generates around what it believes the vehicle is, therefore if the dimension of it is not correct, then the position will be incorrectly positioned as well.

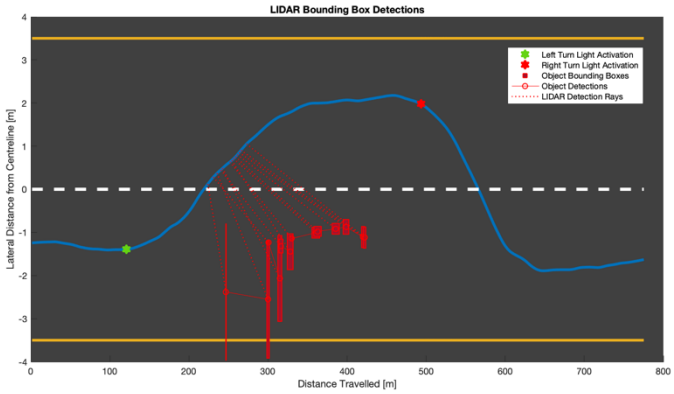

Figure 24. Object's bounding boxes detected directly by the LIDAR.

Thus, a correction is required. This is done by correcting the bounding box dimensions for each of the involved points and to correct the position of the boxes' centres. The approach adopted is rather simple:

a. Find the maximum values for both length and width present in all the bounding boxes detected. A maximum limit of the width is established, to not consider values larger than 0.9 times the lane width (around 3.15 m) in width, which may be due to errors in the measurement or another passing object like a sign which distorts the values.
b. Filter out detections which bounding boxes don't comply with both minimum length and width requirements.

- Minimum Length = Max Length / 5
- Minimum Width = Max Width / 5

c. Recalculate the Maximums, since one of the previous maximums may have not complied with one of requisites described in step b. Then it is not necessary to redo step b.
d. The lateral position and width of the bounding boxes is corrected (Figure 25).

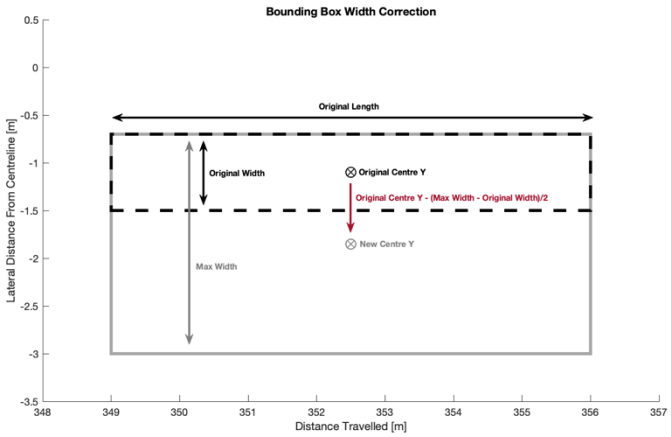

Figure 25. Correction of the bounding box's length and centre's longitudinal position

e. The longitudinal position of the bounding boxes is corrected as follows:

- When the object detection's index is before that of the object with the max width (Figure 26).

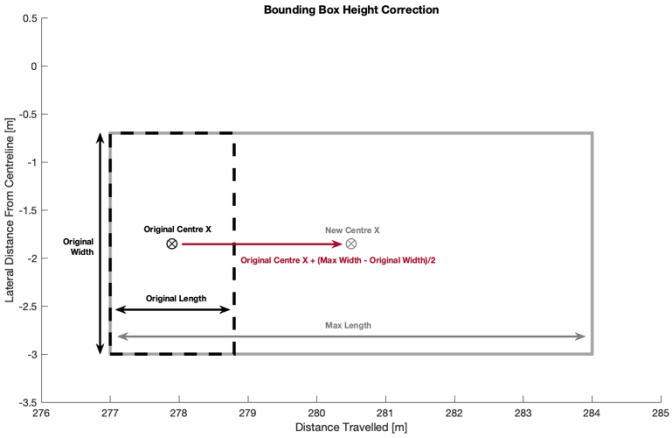

Figure 26. Correction of the bounding box's length and centre's longitudinal position.

- When the object detection's index is after that of the object with the max width (Figure 27).

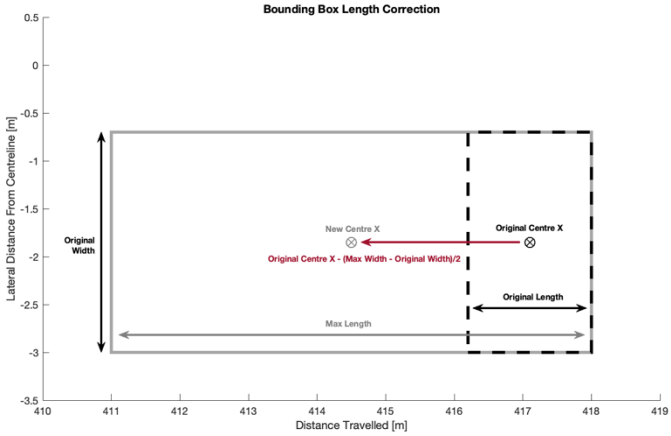

Figure 27. Correction of the bounding box's length and centre's longitudinal position.

Once the points and their respective bounding boxes have been corrected, the detection centres end up looking like what is displayed in Figure 28, and in Figure 29 the bounding boxes have been appended, as well.

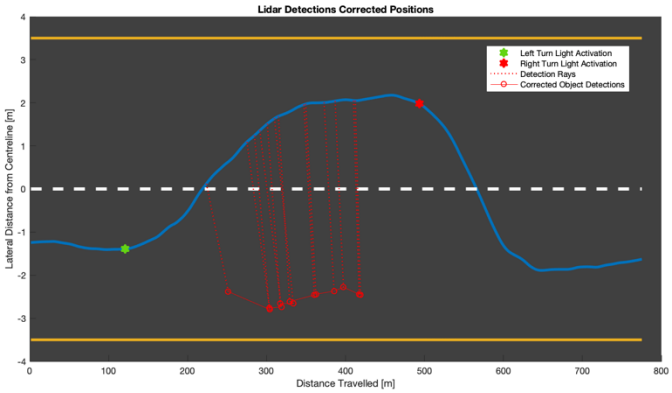

Figure 28. Correction of Detection Positions.

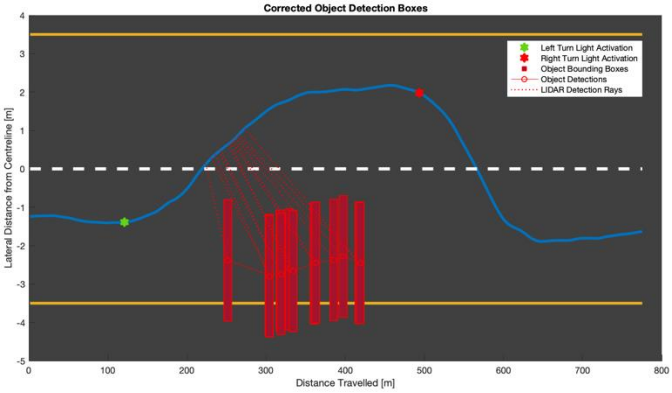

Figure 29. Object's bounding boxes corrected in dimensions and position.

7. *Estimate the overtaken vehicle's trajectory.*
   Since the correction still produces noisy results and does not provide the entire time history of the overtaken vehicle, it becomes necessary to estimate a smooth version trajectory of the entire time interval being analysed. It is clear that this will not represent an unimpeachable reality of the actual event, but it serves as a close enough approximation.

   The trajectory is approximated through a simple multistep process:
   a. Linear interpolation between the detected points in order to estimate the longitudinal position of the vehicle in the intermediate, non-detected, timesteps, and update of the longitudinal position array.
   b. Smoothening of the longitudinal position array, with a second-degree polynomial least squares model local regression model which excludes outliers.
   c. Partial-flattening of the obtained trajectory in order to reduce the amplitude of lateral movement variations if the lateral position array due to the LIDAR position extraction algorithm.

d. Smoothening of the lateral position array with a second-degree polynomial least squares model local regression model which excludes outliers.

An example of the resulting trajectory can be seen in Figure 30.

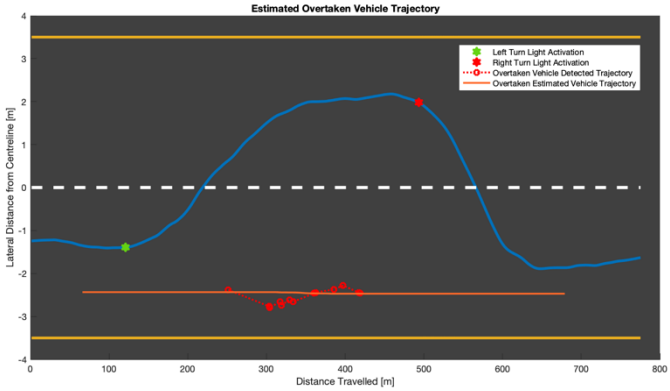

Figure 30. Object's bounding boxes corrected in dimensions, position and detection time.

8. Estimate vehicle velocity and distances from the ego car.

The overtaken vehicle velocity is simply derived from the positional arrays and smoothed in order to filter out unwanted noise (Figure 31).

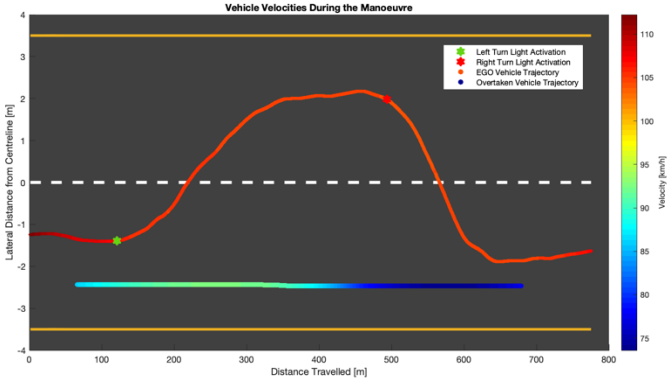

Figure 31. Velocity evolution along the trajectory of both vehicles.

Not only the distance between the ego vehicle centre (Figure 32) and that of the overtaken one is needed, but also an array containing the distance between the ego vehicle's front bumper and the rear of the overtaken car is created, as well as one between the ego's rear bumper and the overtaken vehicle's frontal section. These allows to find the exact moment when the ego car has surpassed the other vehicle, or the free distance between both vehicles at the moment the driver decides to activate the left turn signal.

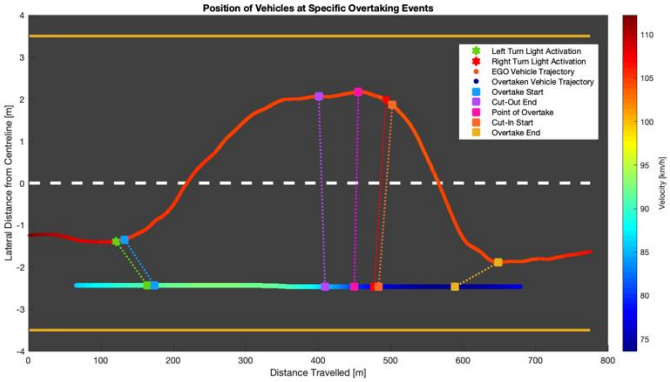

Figure 32. Position the overtaken vehicle's detections with respect to the EGO Vehicle.

9. LIDAR data compression

As done with the CAN data previously, the LIDAR information is compressed to the set compression rate of five, so the amount of data becomes smaller and it is easier to handle afterwards.

After the entire procedure has been completed and the overtaken vehicles and their respective information is known, an array structure will be generated with includes all the important information relevant to each of the overtaken vehicles. The informations included are reported in Table 6.

Table 6. Information relative to each overtaken vehicle

Variable	Description
Position (Longitudinal, Lateral)	Absolute position of the object at every timestep.
Velocity (Longitudinal, Lateral)	Absolute and relative velocities of the object at every timestep.
Distance from Ego Car (3 arrays)	Distance between the overtaken vehicle and the ego vehicle, measured from three different sets of reference points at every timestep.
Distance at specific events	Distance between the overtaken vehicle and the ego vehicle at points such as: left turn light activation, overtake start, cut-out end, right turn light activation, cut-in start, overtake end.
Index of specific events	Indexes referring to point where the ego car catches the overtaken car, exact moment of overtake, etc.

An example of the result where the ego vehicle is overtaking a truck is shown in Figure 33.

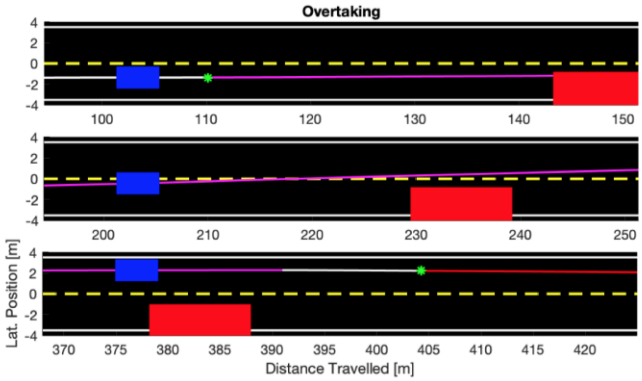

Figure 33. Ego vehicle overtaking a truck.

Calculation and Extraction of Manoeuvre's Parameters of Interest

Once the manoeuvre trajectory has been detected, the information of the vehicles which take part in the overtake are known, and the time of the important events are known, then it is time to extract all the information of interest and store it in an organized structure array. This procedure consists of a series of steps, which include isolating the effects the overtake has on dynamics, extracting the variables of interest and performing specific calculations to extract statistical data from them, not only for the entire manoeuvre but also specific to each subsection of it, and finally store all of it into the corresponding struct.

To better compare the different overtakes, it was decided to isolate the effects the manoeuvre had on vehicle dynamics from those of the road. By doing so, a manoeuvre performed in a curved section and one done on a relatively straight path belonging to a similar scenario, can be analysed and it could be observed how the curvature may influence or not the different values. It is accomplished by observing the values prior and posterior to the overtake for each of the variables referring to vehicle dynamics, and estimating the values referring to the timeframe of the overtake through an auto-regressive function, which is able to comprehend the tendency of the signals from the first section and predict their evolution and how it developed into the signal of the final section. Theses recreated signals are then subtracted from the original CAN Bus values, leaving only those due to the overtake itself, which are then stored in a new set of variables, while the values due to the road are stored in another set of variables, as well. The results can be observed in Figure 34.

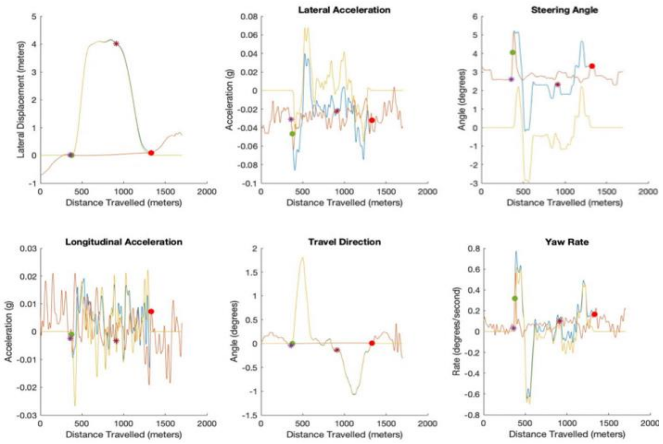

Figure 34. Isolation of overtaking effects on vehicle dynamics.

Table 7 shows the variables of interest and the source from where the raw information came from to obtain them.

Table 7. Manoeuvre parameters and variables of interest.

Variable	**Definition**	**Source**	**Extra Parameters**
Time to Completion	Time it takes the vehicle to travel a specific section of the trajectory.	CAN Bus	-
Distance Travelled	Distance covered alongside the longitudinal axis of the road.	CAN + Camera	-
Lateral Displacement	Position of the vehicle with respect to the centreline of the road.	Camera	-
Lateral Velocity	Velocity with which the vehicles moves toward or away from the centreline.	Camera	Mean Minimum Maximum Peak to Peak
Lateral Acceleration	Acceleration of the vehicles as it moves toward or away from the centreline.	CAN Bus	
Lateral Jerk	Rate of change of acceleration of the vehicles as it moves toward or away from the centreline.	CAN Bus	
Longitudinal Velocity	Velocity with which the car moves parallel to the vehicle centreline.	CAN Bus	
Longitudinal Acceleration	Acceleration of the vehicle parallel to the centreline.	CAN Bus	
Longitudinal Jerk	Rate of change of acceleration of the vehicle parallel to the centreline.	CAN Bus	
Smoothness	Sum of all discomfort issue happening during the manoeuvre	CAN Bus	
Yaw angle	Angle between the direction the vehicle is heading towards and the longitudinal trajectory of the vehicle.	Camera	
Steering Angle	Angular position of the steering wheel.	CAN Bus	
Important Events	Indexes of important events (overtake limits, cut-out, cut-in, vehicle overtaken, etc.)	All	-

The LIDAR information previously extracted is appended to this set of information, to have the complete set of information regarding the overtaking manoeuvre.

Draft of Valid Manoeuvres

Not all manoeuvres may be of interest or valid for this study, so these need to be discarded and only those which are of interest are then exported and will be part of the final database. One of the main reasons not to include certain sets of manoeuvres is that these might certain values or characteristics which represent an important outlier and will negatively affect the results obtained and may lead to erroneous results. The following is a list of minimum conditions which a manoeuvre must satisfy in order to be considered valid:

$$Number\ of\ Overtaken\ Vehicles = 1 \quad (20)$$

$$15s \leq Overtaking\ length \leq 90s \quad (21)$$

All the valid manoeuvres are stored in a struct which later will become the overall general database of manoeuvres. Each struct entry refers to an individual manoeuvre, and contains the information that characterises and describes it in its entirety as well as another participating vehicle

Manoeuvre Classification

Depending on the specific case or behaviour that it is intended on being studied, the manoeuvres can be classified in different ways. The current approach is that the system is able to automatically generate different statistical scenarios based on the data present in the data relative to a specific set of variables that characterize and differentiate each of these, and then is able to classify the manoeuvres and place them in the corresponding bin and have them as optimally distributed as possible.

Scenario Parameter Selection

Since the objective is to study human driving behaviour, the parameters selected to classify the manoeuvres, should be analogous to the decision process the driver does when overtaking a car. Therefore, the classifier is built to work similarly to a decision tree, where each level of the tree represents a parameter and the path followed is dictated by the value of that parameter in that given moment. Consequently, the order in which parameters are analysed is extremely important, since it dictates how the data is sub-divided/classified.

Also, the scenarios are related to a given macro-scenario, so for example, the moment you decide to signal the others that you will overtake. For this case, the driver would have checked his/her speed, then the speed of the other car, then the distance between them, the lateral position of him and the other vehicle, and then the vehicle type to decide that it was the correct moment to announce he would perform an overtake. This type of thought process has to be performed when studying any type of macro scenario.

Configuring Scenario Decision Tree

In a conventional decision tree, at each node which represents the end of one of the tree's branches, said parent branch is divided into two separate branches, where each child branch represents a variable classification, either a true or false, yes or no, higher and lower than a value type setup. This case called for a more elaborate approach, where for each "level" the number of children nodes (branch divisions) could be configured manually. At the end the number of total possible end nodes will be dictated by:

$$N_{levels} = Number\ of\ Levels \quad (22)$$

$$Divisions_i = Number\ of\ Divisions\ for\ Level_i \quad (23)$$

$$Possible\ N°\ of\ Nodes = \prod_1^{N_{levels}} Divisions_i \quad (24)$$

This approach also takes a different approach on how the intervals that dictate each branch are set. Normally, these are set beforehand, but this model was designed so as to follow a statistical grouping (branch division) based on percentiles. This way more homogenous groupings could be achieved when the data distribution is not known beforehand, and even less on how the distribution looks for every sub group of data that arrives to a node prior to partitioning. The great advantage of using percentiles, is that even though for a specific node, it has been requested that it has to be divided into three sub-sets, if the model does not find it statistically fit to do so, it will divide the data into only two or even one sub-set. For example, if it receives an array of all equal values and it has received an instruction of creating two different groups of data, it understands it is not possible, and will only generate one, which includes all the data.

This approach leads to a number of end nodes equal or smaller than the total possible number of nodes, without generating empty ones. Since end nodes are essentially the different scenarios, the approach effectively creates as many different scenarios as it statistically finds possible based on the initial set of constraints (variables used to classify the data and into how many intervals divide the data corresponding to the analysed variable).

Classifying Manoeuvres

All the manoeuvres are stacked together in a table which includes the variables which make up the scenario decision tree, which is then inputted into the classifier. The data, hence the manoeuvres, are passed through each level of the configured decision tree. Depending on how the scenario decision tree was configured, the system will divide the manoeuvres arriving at each node in the amount of bins requested, or less if it finds it statistically more appropriate, using the percentile segregation method explained previously, until all the manoeuvres are classified into the proper end nodes, which will end up representing the final scenarios.

The classification algorithm outputs the scenario number for each of the manoeuvres and saves it into their respective structs. It also outputs a datasheet describing each of the generating scenarios, meaning the interval limits for each of the classifying variables. It also describes the number of manoeuvres present in each scenario. For example, for Scenario n, the table describing the variable limits could look like Table 8.

Table 8. Example of variable ranges making up a hypothetical Scenario n.

Classifying Variable	**Lower Limit**	**Upper Limit**
Ego Vehicle Speed [km/h]	123.58	126.32
⋮	⋮	⋮
Lateral Position [m]	-1.53	-0.29

Database Generation

Once all the manoeuvres have been classified, it is then possible to generate the final manoeuvre database which can be used for studying how different scenario conditions may alter human driving behaviour and how these manoeuvres after vehicle dynamics and perceived comfort. A database not colligated to scenario study, but towards the study of individual manoeuvre is created alongside it, so it can be destined towards other types of studies as well.

It is important to clarify that more than one type of database aimed towards the first study described can be developed, and it will depend on the specific focus in mind. The first focus would be to compare scenarios to each other, by contrasting their statistical data or comparing directly standard manoeuvre representing the average manoeuvre of each of the scenarios. The second focus would be that of comparing all the manoeuvres belonging to a given scenario in order to analyse its spectra. For each focus, the database adopts a specific structure tailored to it.

The database design selected for the first focus, and main focus of this study, which is to compare scenarios in order to understand how different parameters may affect human-driving behaviour, was that of a table compressed of columns representing the characteristic or statistical parameters of interest and each row symbolizes a different, meaning that each cell holds the statistical value of a parameter of interest of the respective scenario.

For the second point of focus, since the manoeuvres are being compared to each other inside a scenario, it can take a similar approach as the previous case, or rather, something very similar to what is done when building a database used to compare all the manoeuvres. Basically, a MATLAB array structure is extracted from the structure outputted from the classification process (it was generated prior to that but was updated after the classification, so in essence this new database is a child of the original one. This type of design is selected when it is of interest to analyse and/or compare the manoeuvres' time histories, meaning how the velocity is at a given point, or where does the lateral acceleration occur at, and why it might differ between different manoeuvres, similar to a case to case study.

Then, the selection of the variables and parameters which will be compared and analysed between the different scenarios depends on what is required and the objective of the study. In a case study like this one, where comfort is of interest, variables such as lateral acceleration, jerk, smoothness, lateral displacement, time to completion, to name a few, have all been deemed relevant since they describe well the perceived movement of the vehicle alongside the trajectory. In a future development, in which these values will be compared between the different scenarios alongside their dispersion, will allow to understand how the conditions may alter the driver's aggressiveness, for example.

It is up to the person studying the manoeuvres which manoeuvres to include as well, meaning that even though some of them might fall into a given scenario, who is performing the analysis may want to exclude them due to a reason, like a max speed criterion, for example.

When a table is developed, to be used to perform a direct comparison between the different scenarios, a statistical table is generated alongside it. This table in reality is a dispersion analysis table or, more specifically, a percentile analysis table. In contrast to the main table, which for each scenario variable it sets its value as the average of that variable's value along of all the manoeuvres belonging to it, this also returns the percentiles values requested. For example, the user could ask to know percentiles 10, 25, 50, 75 and 90, which would allow to study the dispersion of the manoeuvre data stored inside this given scenario. This becomes specially when comparing two or more scenarios where a specific parameter is being compared and it may seem both have the same average values, which could draw the user to conclude the variable does not affect either scenario differently, but then by comparing the dispersion it is observed, it may happen that for one scenario the dispersion is minimum, so maybe a high correlation exists, and for the other(s) a high dispersion may indicate there is no clear correlation between the scenario and manoeuvre behaviour. A spider plot

scheme has been designed to take special advantage of this percentile analysis table by being able to plot a multi-ring spider web, where each ring represents a percentile range, and each vertex is the percentile value of a respective variable.

The databases generated are exported as a .mat file but can be exported as csv file as well if to be used with Microsoft Excel, Python, R, or another software. Saving a database in a *.mat* extension (a binary data container used by MATLAB) allows it to be easily be loaded into MATLAB to use it in a data analysis script, like the spider plot scheme described previously.

Conclusions and future developments

The methodology itself proved to be very robust and is able reduce immensely the required time and the complexity of analysing a large set of experimental files compared to if it was performed manually. It allowed to automatically traverse all the data files, extract the manoeuvres (and all their respective information of interest), and compile a set of databases which could be used for data analysing. It essentially transforms loads of data into usable information. The methodology is not restricted to a defined number of files, but rather it can process as many as requested and still output *clean* and *elegant* databases, if they have the same basic structure as the ones the algorithm was design around.

In an initial iteration of the software over 300 manoeuvres were found inside the 14 experiments analysed for one of the vehicles, but not all the manoeuvres were acceptable or of interest to the study. Various tweaks have been made since in order to improve the performance, robustness, and the selection process by performing many iterations. As of the current state, it can overcome visually demanding tasks, such as detecting lane markings in shadows, discriminating other objects such as vehicles components, walls, and markings which might assimilate a lane marking and not consider them as so. It is also able to detect most lane changes, even in some visually harsh conditions with noise or other distracting elements which tend to confuse the perceptron algorithm. The LIDAR processing algorithm has been modified so that its sliding windows mechanism adapts to the travelling speed and direction of travel, so it can be more effective in selecting correct detections and leaving out outliers. Other many modifications have been done alongside this, through the various iterations, but the overall process has remained virtually the same. A final number of outputted valid overtakes is yet to be defined, but it is expected to be around 170-220 different manoeuvres once all the parameters and tolerances become definitive, so between 13 to 18 manoeuvres per 20 to 30-minute experimental run in average.

Having all the databases generated, it is then possible to perform the scenario analysis and comparisons describe previously. As a continuation of this work, a statistical analysis will be performed on the data stored in the generated databases. Already two tools have been developed to do so:

1. *Spider Plot Analysis*
   As mentioned previously, a spider plot scheme was developed to easily compare in a visual manner multivariable data (the scenarios in this case). It has the advantage that inside a very compact design it can display simultaneously a multitude of parameters at the same time. Not only that, but it can also display different percentiles values of the different variables making up the data, also in simultaneous. The basic principle is that a polygon is drawn for each scenario with as many vertices as variables chosen for the analysis. The polygon itself is composed of rings, per say, where there is an innermost ring, an outermost ring (this being the outer border of the polygon), and rings in between. Each ring represents a percentile of the data, so if it has five rings, this would represent percentiles 0, 20, 40, 60, 80, 100, but these percentiles englobe the data of all the scenarios being compared. This means, that the lines connecting each of the innermost ring outer most ring's vertices represent a normalized scale of the values of each of the variables being compared, regarding the entire data, not to one scenario. Then, for a specific scenario's variable, its value will lie alongside the line respective to that variable, representing how it is in relation to the max and minimum of the compared set.

   It was built in such a way that it could be very flexible, easily allowing to choose which variables to display, as well as which scenarios to compare and which not to. It could happen that for a future development a dashboard could be built around it, so it does not require a manual input of the database's table, variables and scenarios inside a script, but rather through buttons and a selection menu.

2. *Manoeuvre Dashboard Study*
   To study manoeuvres belonging to a specific scenario, or any set of manoeuvres as well, a dashboard was developed using MATLAB's App Designer. It is able to interpret the database composed of MATLAB array structs and visualize time history information of many variables at once. For example, it can plot the vehicles trajectory, as well as the speed evolution along it, then animate the vehicle traversing along it and displaying through gauges the values of different parameters (ex. Lateral acceleration) at the respective vehicle position, allowing to visually observe how they variate at each timestep (Figure 35).

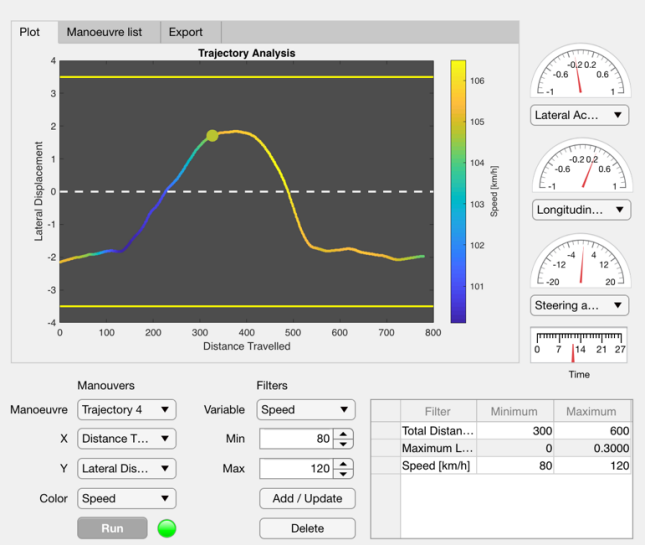

Figure 35. Manoeuvre-Analysis Dashboard.

The manoeuvre scenario database can also be used to train different machine learning algorithms which could be able to classify new overtakes into the generated scenarios so as to predict general tendencies without requiring to process them in their entirety. Another approach would be that from experimental data regarding a specific scenario, these are able to generate a standard manoeuvre optimized for comfort which could be implemented in the development of autonomous vehicles, for example.

In general terms, the overall approach used to study the overtaking manoeuvres, which involves fusing the data from different sensors and sources to isolate them and extract their relevant information, is applicable and scalable towards studying other types of manoeuvres as well. Future work could include developing a more flexible and modular framework which can be adapted to study various types of manoeuvres and not only overtakes. This would require an almost complete rebuild of the current program, practically a reconstruction from the ground up, but many of the algorithms already implemented or at least the working principles these are built around can be recycled or reconditioned to work in this new framework. It should be built around a general common structure or backbone (a set of modules that are fixed, like data handling, data handling function, exporting modules, etc.) and depending on the type of manoeuvre and/or scenario being studied, certain modules are added (if the modules are non-existent, can be developed following the framework structure) so as to complete the program's body structure. This would allow it to study and process any kind of manoeuvre data and to generate a tailored database related to it without having to construct a new data processing methodology in its entirety.

References

1. Carello, M., Ferraris, A., Airale, A., Fuentes, F.: City Vehicle XAM 2.0: Design and Optimization of its Plug-In E-REV Powertrain. SAE International Congress, Detroit (Michigan) 8-10 April, pp. 11, (2014), DOI 10.4271/2014-01-1822.
2. Carello M., Brusaglino G., Razzetti M., Carlucci A.P., Doria A., Onder C.H., "New technologies demonstrated at the Formula Electric and Hybrid Italy 2008", 24th International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium and exhibition EVS24, Stavanger (Norway) 13-16 May 2009, World Electric Vehicle Journal, Vol. 3, Issue 1, 2009, ISSN: 20326653.
3. Filippo N.; Carello M.; D'Auria M.; Marcello A., "Optimization of IDRApegasus: Fuel Cell Hydrogen Vehicle" SAE International Congress, Michigan (USA) 16-18 April, pp. 9, 2013, DOI: 10.4271/2013-010964.
4. Ferraris, A.; Xu, S.; Airale, A.G.; Carello, M., "Design and optimization of XAM 2.0 plug-in powertrain", International Journal of Vehicle Performance, Inderscience, pp. 25, 2017, Vol. 3, ISSN: 1745-3208, DOI: 10.1504/IJVP.2017.10004910.
5. Carello M., Filippo N., D'Ippolito R. "Performance optimization for the XAM Hybrid Electric Vehicle prototype" SAE 2012 World Congress and Exhibition, Detroit (MI) April, 2012. DOI: 10.4271/2012-01-0773
6. Carello, M., Ferraris A., Airale A., and Fuentes F. "City Vehicle XAM 2.0: Design and Optimization of its Plug-in E-REV Powertrain." SAE 2014 World Congress and Exhibition, Detroit (MI) April, 2014. doi:10.4271/2014-01-1822.
7. Ferraris, A., Airale A. G., Messana A., Xu S., and Carello M. "The Regenerative Braking for a L7E Range Extender Hybrid Vehicle." 2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe, 16 October, 2018. doi:10.1109/EEEIC.2018.8494000
8. Carello M. and Messana A., "IDRApegasus: a fuel-cell prototype for 3000 km/L", Computer-Aided Design and Applications, CAD Solutions, LLC, pp. 15, 2015, Vol. 11 (a), ISSN: 1686-4360.
9. de Carvalho Pinheiro, H., Messana A., Sisca L., Ferraris A., Airale A. G., and Carello M. "Computational Analysis of Body Stiffness Influence on the Dynamics of Light Commercial Vehicles". Mechanisms and Machine Science, Vol. 73, 2019. doi: 10.1007/978-3-030-20131-9_307
10. Slim, Rayan, Amer Sharaf, Jad Slim, and Sarman Tanveer. "The Complete Self-Driving Car Course." Accessed December 15, 2018. https://www.udemy.com/course/applied-deep-learningtm-the-complete-self-driving-car-course/.
11. Hult, R. and Tabar, R.S., "Path Planning for Highly Automated Vehicles," Master's Thesis, Chalmers University of Technology, Gothenburg, Sweden, 2013.

12. Wei, J., Dolan, J.M., and Litkouhi, B., “A prediction- and cost function-based algorithm for robust autonomous freeway driving,” 2010 IEEE Intelligent Vehicles Symposium, 512–517, 2010, doi:10.1109/IVS.2010.5547988.
13. Do, Q.H., Tehrani, H., Mita, S., Egawa, M., Muto, K., and Yoneda, K., “Human Drivers Based Active-Passive Model for Automated Lane Change,” IEEE Intelligent Transportation Systems Magazine 9(1):42–56, 2017, doi:10.1109/MITS.2016.2613913.
14. Altché, F., Polack, P., and Fortelle, A. de la, “High-Speed Trajectory Planning for Autonomous Vehicles Using a Simple Dynamic Model,” ArXiv:1704.01003 [Cs], 2017.
15. Lim, W., Lee, S., Sunwoo, M., and Jo, K., “Hierarchical Trajectory Planning of an Autonomous Car Based on the Integration of a Sampling and an Optimization Method,” IEEE Transactions on Intelligent Transportation Systems 19(2):613–626, 2018, doi:10.1109/TITS.2017.2756099.
16. de Carvalho Pinheiro, H., Messana A., Sisca L., Ferraris A., Airale A. G., and Carello M. “Torque Vectoring in Electric Vehicles with in-Wheel Motors”. Mechanisms and Machine Science, Vol. 73, 2019. doi:10.1007/978-3-030-20131-9_308
17. Ferraris, A., de Carvalho Pinheiro H., Galanzino E., Airale A. G., and Carello M. "All-Wheel Drive Electric Vehicle Performance Optimization: From Modelling to Subjective Evaluation on a Static Simulator." Electric Vehicles International Conference, EV 2019. Bucharest, Romania, October, 2019. doi:10.1109/EV.2019.8893027
18. Altche, F., Qian, X., and La Fortelle, A. de, “An Algorithm for Supervised Driving of Cooperative Semi-Autonomous Vehicles,” IEEE Transactions on Intelligent Transportation Systems 18(12):3527–3539, 2017, doi:10.1109/TITS.2017.2736532.
19. Zhang S.; Deng W.; Zhao Q.; Sun H.; Litkouhi B.; “Dynamic Trajectory Planning for Vehicle Autonomous Driving”, IEEE International Conference on Systems, Man, and Cybernetics, 13-16 Oct. 2013, doi: 10.1109/SMC.2013.709, Electronic ISBN: 978-1-4799-0652-9.
20. Carello M., Ferraris A., Bucciarelli L., Gabiati G., Data A. “Customer Oriented Vehicle Dynamics Assessment for Autonomous Driving in Highway”. SAE International Congress, Detroit (MI) 9-11 April, pp. 9, (2019), SAE Technical Paper 2019-01-1020, doi:10.4271/2019-01-1020, ISSN 0148-7191.

Acknowledgments

This paper has been developed thanks to a partnership between FCA and Politecnico di Torino within the topic of “Vehicle Dynamics Assessment for Autonomous Driving. The authors would like to thank you, in particular, Silvio Data for its support and suggestions.

Contact Informations

Massimiliana Carello
Politecnico di Torino – Department of Mechanical and Aerospace Engineering
C.so Duca degli Abruzzi, 24 -10129 Torino – Italy
Phone: +39.011.0906946
massimiliana.carello@polito.it

Alessandro Ferraris:
alessandro.ferraris@polito.it

Henrique de Carvalho Pinheiro
henrique.decarvalho@polito.it

Isabella Camuffo
isabella.camuffo@crf.it

Giovanni Gabiati
giovanni.gabiati@crf.it

Definitions/Abbreviations

ADAS	Advanced Driver-Assistance Systems
CAN	Controller Area Network
CI	Cut-In
CO	Cut-Out
FCA	Fiat Chrysler Automobiles
GPS	Global Positioning System

LD	Left lane border distance
LIDAR	Laser Imaging Detection and Ranging
L_{lim}	Lower Limit
LTLA	Left Turn Light Activation
LW	Lane Width
OV	Overtake
RD	Left lane border distance
RD	Right lane border distance
RTLA	Right Turn Light Activation
SAE	Society of Automotive Engineers
U_{lim}	Upper Limit