BACHELOR

Data analysis and modeling of the TU/e Lupo EL

van Brunschot, Matthijs P.H.

Award date:
2022

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mechanical engineering
Dynamics and Control Research Group

Data analysis and modeling of the TU/e Lupo EL

Bachelors final project

Matthijs van Brunschot

Supervisors:
dr. ir. I.J.M. Besselink

DC 2022.059

Eindhoven, July 2022

Abstract

To reduce dependency on oil and fossil fuels the automotive industry is transitioning towards electrification. Battery electric vehicles are getting more attention as they are seen as a good path towards a future with zero local emissions for driving. As Battery electric vehicles are relatively new on the consumer market there are still a lot of opportunities for further development of their technology. To gain a better insight in these opportunities the dynamics and control group of the technical university of Eindhoven developed a battery powered electric vehicle as a research platform. A lot of attention was given to data acquisition in the development of this vehicle. Therefore, the vehicle is equipped with an extensive sensor set that covers the powertrain, vehicle dynamics and other aspect of the vehicle. Many test have been executed over the year and tens of gigabytes of measurement data have been collected.
The goal of this project is to streamline the processing of the all the collected measurements. Based on exciting software a MATLAB script is developed that processes the collected measurements into a usable format in MATLAB. The existing software is made more robust so it can handle processing of large amount of measurement files at once. Also functions have been added to cover exceptions such as missing GPS data by using data from a secondary data acquisition system, an Android tablet. Metrics have been developed to characterize the processed measurement files, such that it is more easy to identify what data is available in a certain file. These metrics are based on driving scenario's recognized in the measurement data. To give further insight into what data is available in a certain measurement file a fingerprint tool is developed. This tool presents key numbers of all trips in a table. Alongside this table the route of the trip is plotted on an interactive map. By clicking on a key number of a trip the plotted route will indicate the magnitude of the key number throughout the trip.
This report also includes analysis of the vehicle dynamics of the Lupo EL. A model for wheelspeeds is derived and verified with measurement data. Furthermore, a single track vehicle model is derived and verified with measurement data.

Contents

Chapter 1

Introduction

The automotive industry is shifting towards a more sustainable way of transportation. In particular the battery powered electrical vehicle (BEV) has seen a major increase in popularity. Because BEV's are reasonably new on the consumer market, there are still a lot of opportunities for further development. To get a better insight in these opportunities in the development of battery powered electric vehicles, the Dynamics and Control group of the Eindhoven University of Technology has developed the Lupo EL electric vehicle as a research platform for electric mobility. This research platform gives future engineers a great opportunity to learn about the development of a battery powered electric vehicle in their study and gain knowledge and interest, so they can built a greener future later on in their career. Data collected during the development and testing of the vehicle can be used as lecture material therefor special attention is given to data acquisition features. This thesis gives an overview how data acquisition is done in the TU/e Lupo EL. Furthermore the analysis of the acquired data is streamlined.

1.1 Project Goals

The project goals are as follows:

- Streamline the processing of the CAN logger measurement files to a usable format in MATLAB. Existing software can be used as a starting point, but must be made more robust to cover exceptions (e.g. missing GPS data) and able to handle all available measurement files.
- Develop a method/metrics to characterize a trip, such that it is more easy to identify what data is available within a certain measurement file.
- Develop a simple vehicle model that is able to reproduce the relations between various measurement signals that uses at least vehicle speed and steering wheel as inputs.

1.2 Structure of the Report

This report is organized as follows. Chapter 2 provides a summary of the CAN bus and data acquisition system. Furthermore, it explains in detail how the acquired data is processed into a usable format in MATLAB. Chapter 3 explains how trip data is characterized and elaborates on a tool developed to easily identify what data is available within the different measurement files. Chapter 4 elaborates on the steering system of the vehicle and derivation of simple vehicle models. These vehicle dynamics models are validated using measurement data.

Chapter 2

Data acquisition and processing flow

The design of the CAN bus and data acquisition systems of the TU/e Lupo El are thoroughly researched. Full reports on this subject can be found in the reports [6], [3]. This chapter will give an overview of the CAN bus systems and the data acquisition procces. Furthermore, it will give an overview of how the acquired data is processed into a usable format in MATLAB, for more convenient and clear post-processing of the data collected during various trips.

2.1 CAN bus

The VW Lupo El is equipped with two separate CAN buses for the various components to communicate with each other with minimal interference and maximum redundancy. One CAN bus is used for the original VW Lupo 3L components to exchange and monitor signals such as dashboard indicators, gear lever, airbag, and ABS signals. Various components of the electric power train such as the inverter, charger, and BMS are linked with each other through a second CAN bus. The built-in Android tablet can connect via an OBD port. The Android tablet has the application ”Torque” installed, this software allows to display and log signals from the CAN bus through the OBD connection. A central programmable logic controller (PLC) is connected to the CAN buses and is used as the main controller. Figure 2.1, gives an overview of the electrical systems layout in the vehicle with the two CAN buses indicated.

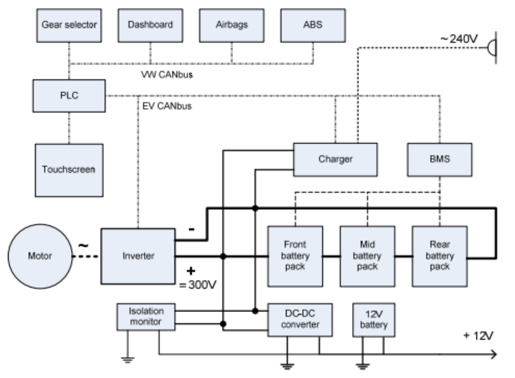


Figure 2.1: Overview system lay-out VW Lupo EL

2.2 Data acquisition

In the aforementioned CAN-bus system lay-out and the rest of the vehicle design, special attention is given to data acquisition. Figure 2.2, shows the layout of the electric power train, including all components connected to the high voltage battery system. The circled U, I, or T symbols are measurement points of respectively voltage, current, and torque measured by the new CAN bus. Next to measurements on the High voltage system also, the majority of the low voltage 12 V system components are instrumented for a more detailed insight into the energy consumption of the vehicle. Besides measurements on the electrical components, the vehicle monitors sensors of the original Lupo 3L CAN bus, for example sensors considering the chassis. These sensors include accelerations, yaw rate, and steering angle. A more detailed overview of signal ID's can be found in Appendix A. The VMS-CAN stand-alone CAN-bus data acquisition system, manufactured by Venamics, is installed for high-speed and reliable logging of CAN-bus data of both the CAN bus connecting all the original VW Lupo components and the CAN bus connecting the electric drive train components. The VMS-CAN CAN-bus data acquisition system uses a memory card containing a configuration file, defining what signals should be logged and stored on the memory card. The configuration file can be created using the VMS-tool for Windows. In the VMS-tool DBC files containing CAN databases can be loaded. The CAN databases contain the different message ID's with subsequent data on the signals and other data, like time frame etc. In the VMS-tool the message ID's and data that is desired to be logged can be selected and stored as the config file. The CAN database of the original VW systems was reversed engineered. Certain signals on CAN bus for the built-in Android tablet are also logged. The "Torque" software running on the Android tablet logs data through the OBD connection. The software also logs data from the tablet itself for example GPS data from the GPS sensor on board of the tablet. The "Torque" application stores the log files as .csv files called Tracklogs.

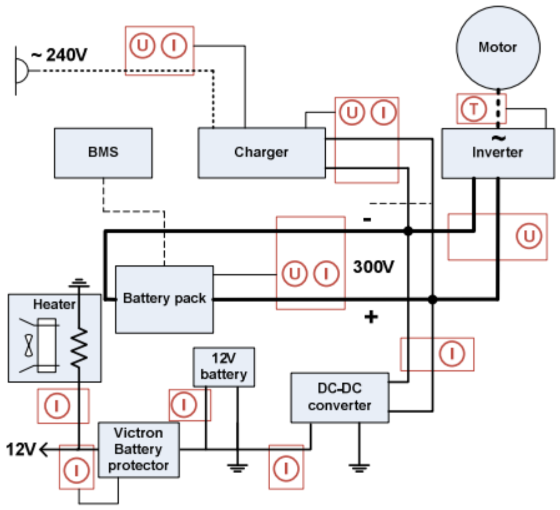


Figure 2.2: High Voltage system layout and measurement points

2.3 Data processing

Data acquired by the VMS data acquisition system of the two main CAN buses is stored as .bin files on a memory card. For convenient post-processing of this data it is preferred to load the data in MATLAB, so a reliable way to export the data in MATLAB is needed. The bin files from the CAN-bus acquisition system contain data from separate CAN buses both, with a variety

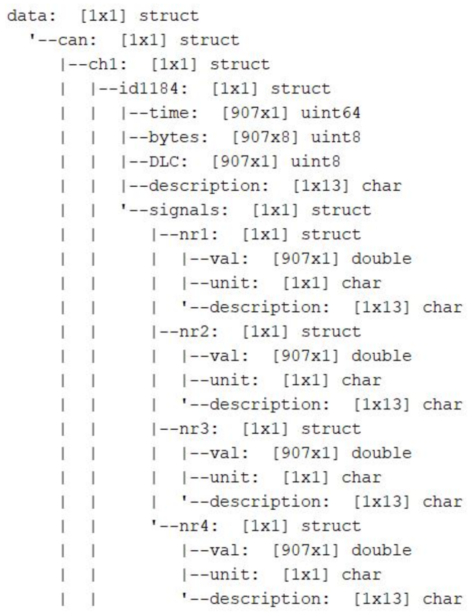
of different sensors all working at different sample frequency's. Therefore all signals need to be resampled and aligned to one generic time base after it is loaded into to MATLAB, so signals can be compared and relevant relations can be found between signals.
The built-in Android tablet also acquires GPS data, as in 2013 and 2014 the GPS signals from the vehicle are not recorded by the VMS CAN-bus acquisition system yet. The GPS data from the tablet can be used to compensate for the missing location data in those years. Again it is important that the right GPS data from the tablet is resampled and aligned with the generic time base of the other signals. Because the data from the Tablet is recorded by a separate system than the other data, the start and stop times of these signals do not necessarily correspond to that of the signals logged by the VMS CAN-bus acquisition system. So not only a reliable way to resample the data needs to be found, but also the right GPS data needs be to macthed and aligned with the VMS data. To streamline the process of going through all these step the MATLAB script *Data_conversionv2*, which is displayed in Appendix B, has been developed. The script goes through the steps:

- Loading data into Matlab.
- Resampling VMS data.
- Combining VMS data and GPS data built-in tablet (if GPS data VMS is not available).
- Adding SRTM elevation data.

The steps will be explained in detail next.

2.3.1 Loading data into MATLAB

The data recorded by the VMS CAN-bus system can be loaded into MATLAB with software provided by the manufacturer of the VMS-CAN system, Venamics. The executable "VMS2MAT.exe" can be called from within MATLAB. This executable will convert the data from the bin files and create a .mat file named VMSlog_YYYYMMDD_HHMMSS.mat, containing a MATLAB structure called "data". An overview of this structure is show in Figure 2.3 for one message ID as an example.



```
data:  [1x1] struct
  '--can:  [1x1] struct
     |--ch1:  [1x1] struct
     |  |--id1184:  [1x1] struct
     |  |  |--time:  [907x1] uint64
     |  |  |--bytes:  [907x8] uint8
     |  |  |--DLC:  [907x1] uint8
     |  |  |--description:  [1x13] char
     |  |  '--signals:  [1x1] struct
     |  |     |--nr1:  [1x1] struct
     |  |     |  |--val:  [907x1] double
     |  |     |  |--unit:  [1x1] char
     |  |     |  '--description:  [1x13] char
     |  |     |--nr2:  [1x1] struct
     |  |     |  |--val:  [907x1] double
     |  |     |  |--unit:  [1x1] char
     |  |     |  '--description:  [1x13] char
     |  |     |--nr3:  [1x1] struct
     |  |     |  |--val:  [907x1] double
     |  |     |  |--unit:  [1x1] char
     |  |     |  '--description:  [1x13] char
     |  |     '--nr4:  [1x1] struct
     |  |        |--val:  [907x1] double
     |  |        |--unit:  [1x1] char
     |  |        '--description:  [1x13] char
```

Figure 2.3: Overview of the MATLAB "data" struct created by VMS2MAT.exe

As can be seen in this overview the total structure "data" contains the substructure "can", in which two substructures for each of the CAN buses can be found. "ch1", channel 1 which correlates to the logged data on the the CAN bus containing the original VW Lupo 3L components. "ch2", channel 2 correlates to the data from the CAN bus containing the electrical power train components. The structures ch1 and ch2 are made up of substructures for each message ID logged as instructed by the config file loaded on the memory card of the VMS CAN-bus system. For an overview of message ID's that are logged see Appendix A. The substructure for a certain message ID contains the time base of all signals present for that message ID as well as the values for the different signals.

2.3.2 Resampling

For convenient post-procession of the exported data with "VMS2MAT.exe" the exported data is resampled, so all signals are aligned with a generic time base and are restructured in a clearer way. This is done by using a combination of the MATLAB functions *VMSconvMAT* and *VMSresample*. The function *VMSconvMAT* first takes a dummy signal to determine the time range of the recorded signals. The dummy is taken by loading the time instances for signal nr1 of message ID 1184 containing ABS data on the front left wheel speed, the min and max of the time recordings are taken to determine the start and stop time of of the recorded signal, these will be used to create a generic time base for all signals. Then the function *VMSconvMAT* analyses the measurements to determine whether the car was driving or charging during the recording. This is done by analysing the front left wheel speed and AC current measurements for a time frame determined with the dummy. If the mean front left wheelspeed is zero and the mean AC current is larger than one for that time frame, the function concludes that the vehicle was charging otherwise the function concludes that the vehicle was driving. This distinction between driving or charging determines the sample frequency used to resample the data. For driving all data will be resampled to a sample frequency of 25 [Hz] and for charging a sample frequency of 5[Hz] is used. This has been done to reduce the datafile size of the charge cycle. The conversion script finally calls the resampling using interpolation the function *VMSresample* resamples each signal using the following options:

- The sampling frequency, 25 [Hz] for a drive cycle and 5[Hz] for a charge cycle.
- Signal Name and Unit.
- Scaling factor, some signals require a certain multiplication in order to be valid. This can be caused by a reverse in direction (-1) or for example the gravitational constant ($g = 9.81$).
- Interpolation Method, for most signals a linear interpolation is used. However, for signals that are either on or off, a 'nearest' interpolation is used.

The resampled signals are stored in structure "m", for driving this structure contains the signals listed in Table 2.1.

Table 2.1: overview ”m” struct

Group	Signal	Unit	Group	Signal	Unit
Chassis	Wheelspeed FL	[km/h]	HV	voltage	[V]
	Wheelspeed FR	[km/h]		current	[A]
	Wheelspeed RL	[km/h]		power	[W]
	Wheelspeed RR	[km/h]		DCDC	[A]
	s	[km]		cell_Vmin	[V]
	a_x	[m/s^2]		cell_Vmin_nr	-
	a_y	[m/s^2]		cell_Vmax	[V]
	yawrate	[deg/s]		cell_Vmax_nr	-
Charger	DC voltage	[V]		SOC	[%]
Dashboard	Odometer	[km]		DOD	[%]
Motor	rpm	[rpm]		nrg_discharge	[kWh]
	torque	[Nm]		nrg_regen	[kWh]
	inverter	struct	LV	voltage	[V]
Controls	steerangle	[deg]		current	[A]
	throttle	struct	Temp	chargeplug	[deg.C]
	brake	struct		coolant	[deg.C]
	ecomode	-		PLC	[deg.C]
	gearlever	-		Charger	[deg.C]
	regenbraking	-		pack1	[deg.C]
	steerangle_raw	[deg]		pack2	[deg.C]
	accelerator	struct		pack3	[deg.C]
	OPD	struct		cell_low	[deg.C]
	CC	struct		cell_high	[deg.C]
Time	time	[s]		cell_avg	[deg.C]

2.3.3 Combining VMS data and GPS data built-in tablet

Having GPS location data combined with measured signals of the vehicle can help interpret the data. GPS data from the Lupo was logged from mid 2014 and on but, for the VMS data of 2013 and the beginning of 2014 there is no GPS data present. A method is developed to combine GPS data from the built-in Android tablet and the VMS data. GPS data and other data recorded by the Android tablet are saved in .csv files named trackLog-YYYY-mmm-DD_HH-MM-SS.csv. The first step in combining VMS data with GPS data from the tablet is to read the .csv files in to MATLAB in a reliable way. The MATLAB function *TorqueconvMAT* reads the .csv files line by line and determines whether the line contains values or column titles, as the Torque application repeats the titles randomly in the log files, which can cause problems when loading the data. Finally the Function *TorqueconvMAT* creates a MATLAB structure ”T” containing all signals logged by the tablet.
The resampled VMS trip data needs to be matched with GPS data from the tablet so, the .csv file containing signals of the same trip needs to found. This is done done in MATLAB by taking the date from the title of the resampled VMS .mat file as a string and converting it to same format as used in the trackLog files. This converted string is used to find all .csv trackLog files recorded on the same day as the VMS file. This narrows down the possible .csv trackLog files that can contain signals of the same trip. Then all trackLog files matching that date are loaded into MATLAB using TorqueconvMAT. For the date matching trackLog files the signals SpeedOBD, longitude and latitude are resampled using the *interp1* function to the same sample frequency as the VMS data so, 25 [Hz]. Then the signal of the wheelspeeds of the VMS data is compared to the SpeedOBD signal of the tablet data using the MATLAB function *finddelay* of the Signal Processing Toolbox toolbox [5]. The *finddelay* function uses the *xcorr* function to determine the cross-correlation between each pair of signals at all possible lags specified by the user. The

normalized cross-correlation between each pair of signals is then calculated. The estimated delay is given by the negative value of the lag, for which the normalized cross-correlation has the largest absolute value. If a delay is found between the measured velocity based on the wheelspeed signals from the VMS data and velocity signal SpeedOBD from the tablet data, the signals SpeedOBD, longitude and latitude are shifted with that delay and stored in the struct m as m.TorqueData. Figure 2.4, shows the steps of resampling and shifting the SpeedOBD signal from the tracklog data and the wheelspeed signal from the VMS data. If no or multiple delays are found by the function finddelay a string containing that information is stored as m.TorqueData.

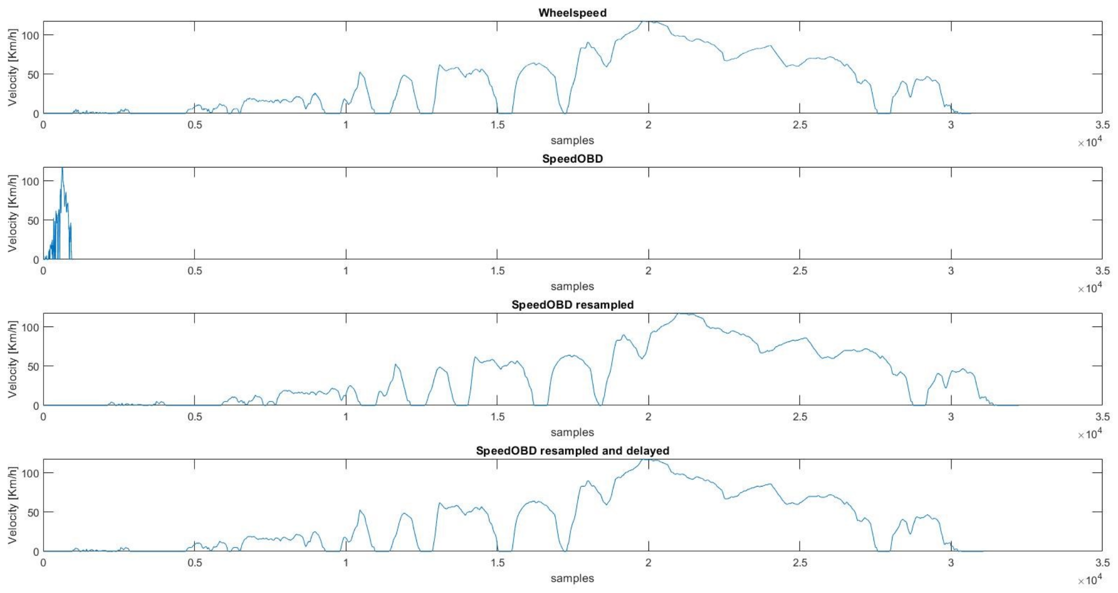


Figure 2.4: Steps aligning speed signals using finddelay

The method of matching signals with finddelay proved to only be reliable if both the tracklog data and VMS data contained data in roughly the same time frame and still resulted in some wrong matches. So, in the data conversion the validity of the alignment of the velocity based on the measured wheelspeed signals from the VMS data and the SpeedOBD signal is checked using the function *Matchcheck*. This function calculates the difference between the signals and if the mean of the difference is within -2 and 2 km/h it is concluded the match is correct.
If it is determined that the match is not correct or no match was found, the function *rematch* is used. This functions tries to match the signals by using the MATLAB function *findsignal* from Signal Processing Toolbox toolbox, The function *findsignal* returns the start and stop indices of a segment of the resampled SpeedOBD tracklog data array, that best matches the VMS wheelspeed data array, signal. The best-matching segment is such that the MATLAB function *dist*, the squared Euclidean distance between the segment and the search array, returns the smallest value. A visual representation of the execution of the function *findsignal* is given in figure 2.5.

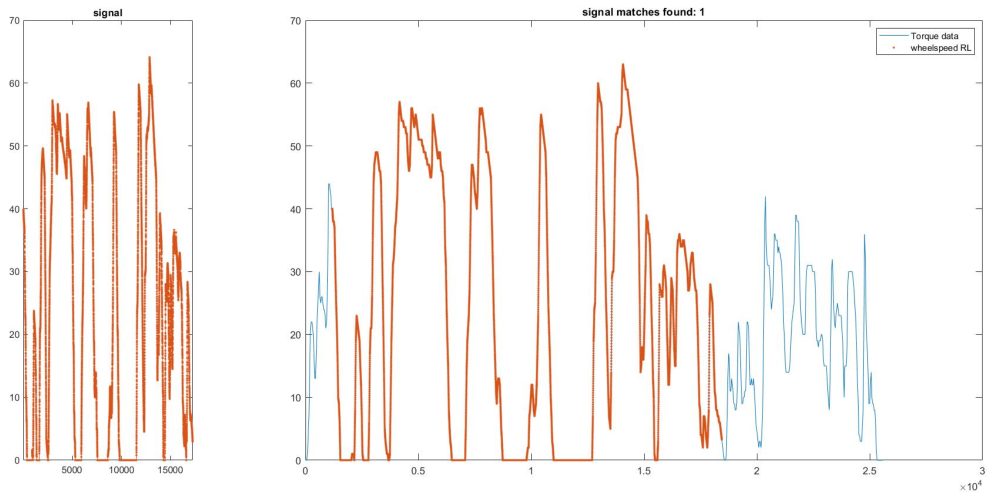


Figure 2.5: aligning speed signals using findsignal

This function is better suited for cases where the tracklog contains data of a larger time frame than the VMS log, but it still may result in false positive matches. So, again the function *Matchcheck* is used to determine if the match is valid. If the match is correct the signals SpeedOBD, latitude and longitude are stored in the substructure m.TorqueData .

2.3.4 Adding SRTM elevation data

As the elevation data of the both the GPS signal from the vehicle and the tablet is not reliable, a different method to add elevation data to the trip logs is used. The MATLAB function *get-SRTMelevation* of the MEC pro toolbox [1] is used to get SRTM elevation data based on the trips recorded latitude and longitude. The function queries and downloads relevant sections of the digital elevation map. The resulting elevation, as function of travelled distance, is filtered by a 3rd order Butterworth low-pass filter with cut-off spatial frequency $\lambda_c = 1/2000\frac{l}{m}$. This data is either stored in trip struct m under m.gps.SRTMelevation if gps signals from the vehicle were available otherwise it is stored under m.Torquedata.SRTMelevation.

Chapter 3

Trip characterization

When dealing with a large database of vehicle trip logs, it is desirable to gain information about the context in which this data was acquired. A basic piece of context, that is not directly contained in the recorded data, is for example whether a trip was recorded in a urban environment or not, that is, if the vehicle was driving on a regular street or on a highway. This chapter presents different possibilities to differentiate driving contexts based on vehicle and location data.
Furthermore, all data recorded is analysed to provide an understanding of the different conditions the vehicle was operated during data acquisition. When reviewing the collection of measurement files there is no simple/quick way to identify what type of test was executed, where it was done, how long it lasted, how much energy was used. This section will elaborate on a method developed to give quick overview of key numbers of a certain log file, combined with visual representation on an interactive map to see where the vehicle is driving.

3.1 Data analysis of all data recorded

Before analyzing data from on a trip level, it is insightful to first analyze all data recorded together. By analyzing all recorded data, an understanding can be gained under what conditions the vehicle was operating, also it gives an insight in how the vehicle typically operated, so special cases where atypical behaviour occurs can be identified.
Using the trip data considered to be valid after the data processing process, some statistics are calculated, see Table 3.1.

Table 3.1: Statistical data valid logs

description	value
number of trips	265
total distance	6036.78 [km]
total amount of electricity (DC)	758.01 [kWh]
average energy usage (DC)	188.4 [Wh/km]
shortest trip	0.00 [km]
longest trip	228.08 [km]
average trip length	22.78 [km]
highest velocity	127.67 [km/h]
average velocity	29.31 [km/h]
highest longitudinal acceleration	6.74 [m/s^2]
highest lateral acceleration	7.65 [m/s^2]
maximum elevation change	85.00 [m]

Considering that this data was gathered over a period of 9 years and the total distance driven is only around 6000 km it can be concluded that the vehicle was not used intensively or data was not logged consistently.

3.1.1 Drive Style

Plotting the longitudinal acceleration as a function of the forward velocity provides a way recognize the aggressiveness of the driver. A binned scatterplot for all the collected for the lupo EL is shown in figure 3.1. As is shown in this figure for the most part the vehicle is driven normally as the bins with the largest amount of data points are further way from the motor power and braking limit. But there are instances where the vehicle was pushed to its limits. This was problably done during acceleration and braking tests.

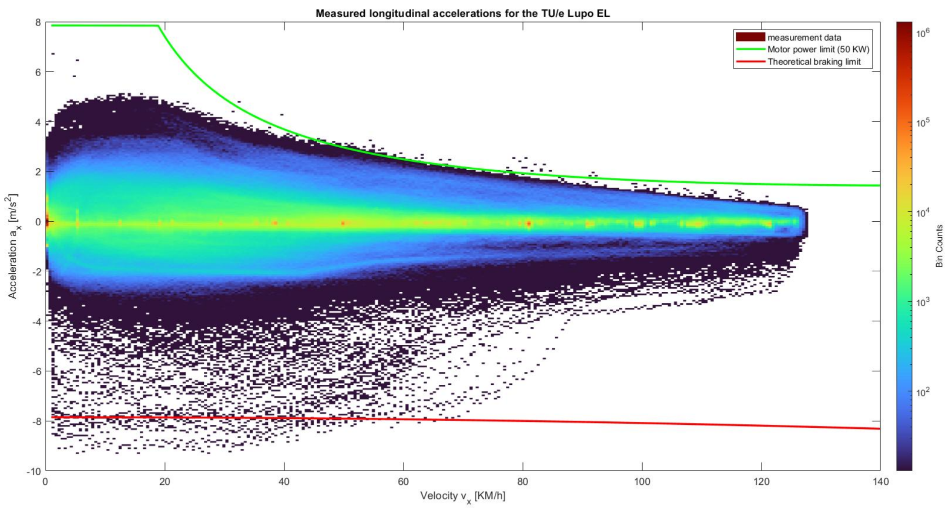


Figure 3.1: Measured acceleration and deceleration on the TU/e Lupo EL

Maximum longitudinal and lateral acceleration cannot occur at the same time, since the tire forces are limited by a friction circle or ellipse. Plotting the maximum achievable combined longitudinal and lateral acceleration results in a so called g-g diagram, shown in Figure 3.2. The g-g diagram can be used to assess driver performance. The graph also shows the physical limits of the vehicle. An “ordinary driver” g-g diagram typically has a cross shape, as also seen in figure 3.2. Furthermore it shows that a higher lateral acceleration was achieved when cornering to the left, this could be because cornering test have been done where was driven on a trajectory with only or mainly left hand turns.

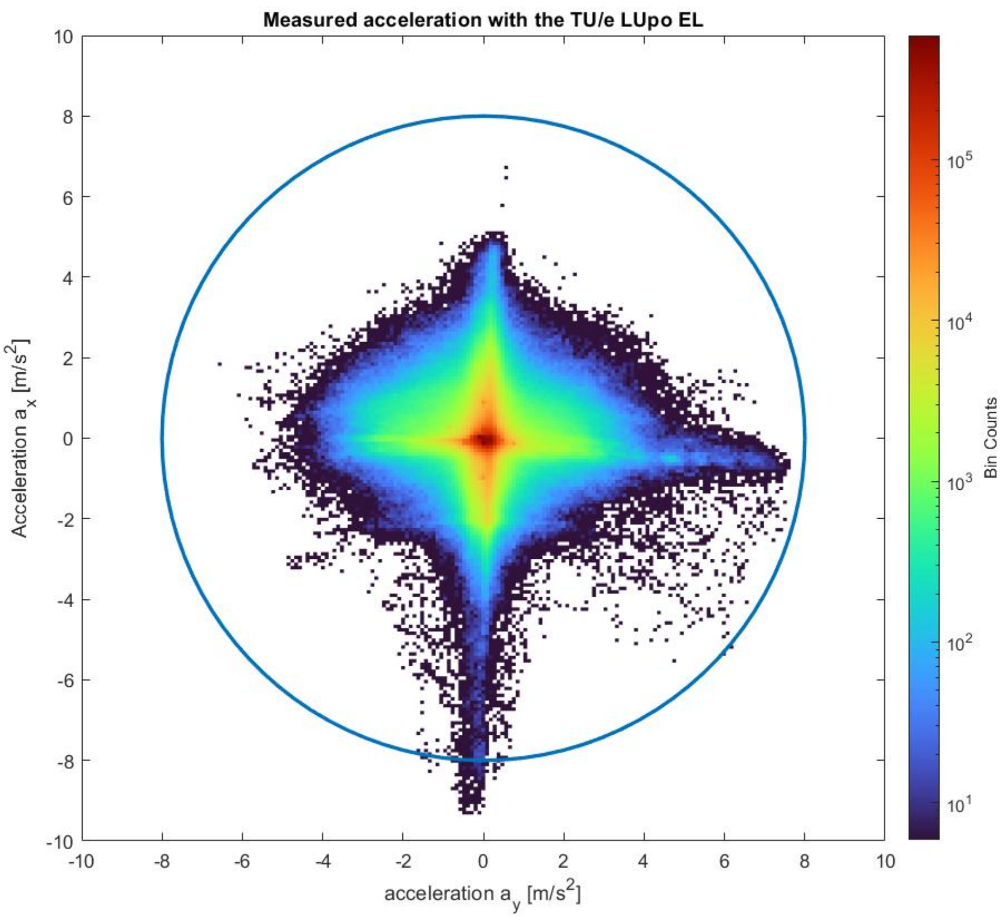


Figure 3.2: Measured acceleration with the TU/e LUpo EL

3.2 Defining metrics trip characteristics

Based on visual evaluation of location data, three distinct driving scenario's can be recognized: Urban trips, Highway trips and trips solely driven on the TU/e campus. By manually selecting a number of trip logs of each of these categories and comparing the vehicle signals of the selected trip notable differences between the categories are sought that can be used as metrics to characterize collected vehicle data.

Based on the values presented 3.2, it is noticeable that there are clear distinctions in the average distance of trips between the three category's. Campus trips are obviously the shortest as the campus has quite limited space to drive around. Furthermore Highway trips are longer, which is also logical as highways are designed to efficiently travel great distances. Energy use is different in the three scenario's. Campus trips have the highest energy consumption, this can attributed to the fact that the roads on the campus are not that long so a lot of accelerating and decelerating has to occur. Urban have a lower energy consumption than highway trips, this is can be attributed to the fact that the average velocity of urban trips is lower, therefore there is less drag from the air.

Another way to distinguish a campus drive is simply by looking at the coordinates as the coordinates of the TU/e campus are known. It is concluded that the best way to distinguish Urban trips from Highway trips is by evaluating the top speed and average speed, in edge cases the trip

Table 3.2: Comparison data distinct driving scenario's

description	Urban	Highway	Campus
Average distance trip [km]	39.26	69.41	1.49
Average energy use [Wh/km]	110	135	185
Average max velocity trips [km/h]	56.5	113.2	15.3
Average velocity trips [km/h]	17.41	64.20	15.28
Max longitudinal acceleration [m/s^2]	4.20	3.36	3.03
Average longitudinal acceleration [m/s^2]	-0.0865	-0.0680	-0.0222
Max absolute Lateral acceleration [m/s^2]	3.76	3.76	2.85
Average Lateral acceleration [m/s^2]	0.0493	0.0808	0.22

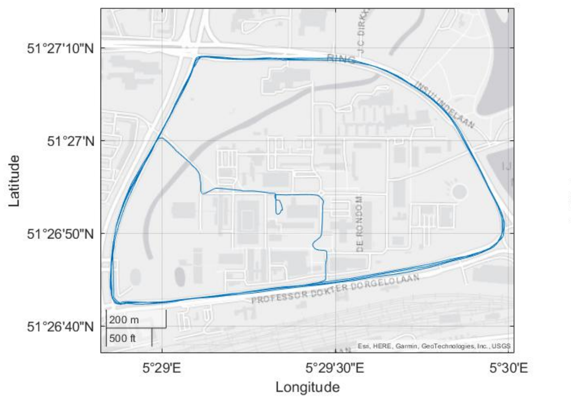
distance can be the deciding factor. The best way to distinguish campus trips is by location data if available otherwise by by low average distance and high average energy use.

3.2.1 implementation of classifications

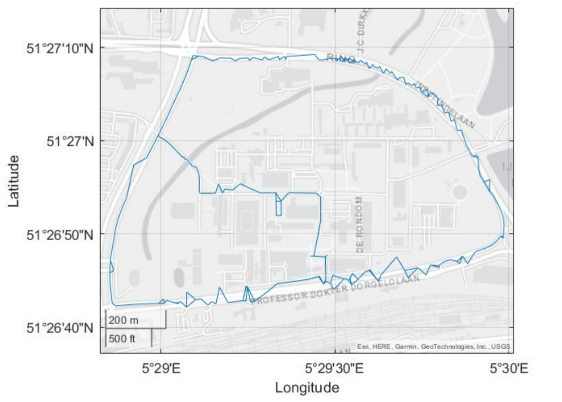
From the comparison of the different driving scenario's a final method to characterise the trip measurement files has been developed. First if location data is present it is determined if the maximum latitude and longitude are below the north east boundary conditions for the campus which are set at 51.45 and 5.50 respectively and the minimum latitude and longitude are over 51.4 and 5.8. Then the average and top speed of the trip are determined. If the average speed is above 50 km/h and the top speed above 95 km/h the trip will be classified as highway. If the average speed is under 50 km/h and the top speed under 95 km/h the trip is classified as Urban. The characterization is implemented in the naming system of the VMS files. The fourth until sixth characters in a files name form its classification, for example "VMScty_yyyymmdd_hhmmss.mat" . The characters 'cty' are used for urban trips, 'hgw' for highway and 'cmp' for campus trips.

3.3 Trip characterization based on Location data

Based on the metrics found in Section 3.2 a little more insight is given into the driving scenario of a trip. However there still is no information on the different road types that was driven on during one trip. Therefor, an attempt was made to use a link to OpenStreetMap based on the recorded location data to acquire more detailed information on the roads that have been driven on during a trip. OpenStreetMap is a geographic database, that includes classification of most roads, streets and paths. It should be able to be used to identify the kinds of roads that have been driven on. For the implementation of identifying road types for a trip log a MATLAB function is developed. This function is derived from a vehicle velocity predicting algorithm, see report [1]. The MATLAB function queries relevant road information from OpenStreetMap. This includes road type, the local speed legislation, traffic sign location and more. This information is queried for an area within the maximum and minimum latitude an longitude recorded in a trip. The information from Openstreetmap is available for specific points in space defined by its latitude, longitude these points are called nodes and have a specific node id. After downloading the OpensStreetMap data the MATLAB script finds the closed nodes to the coordinates from the trip log. However, because the nearest node is not always on the same road as the coordinates of the trip log the route driven gets distorted, this is shown in figure 3.3. Because of this distortion the wrong road data is collected. As of now there is no solotion found to this problem so the trip characterization based on location data still needs be improved further to make it reliable before it can be used.



(a) Input route for characterization using OpenStreetMap



(b) Output route based on nodes OpenStreetMap

Figure 3.3: difference input and output route OpenStreetMap

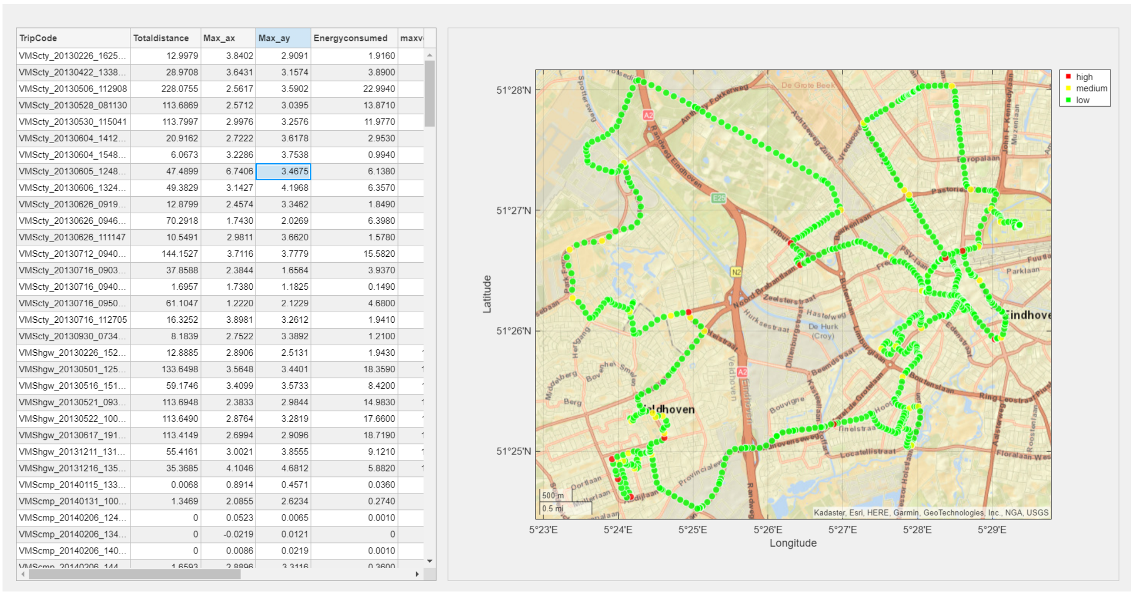
3.4 Fingerprint

Even with the addition of the three driving scenario's, urban, highway and campus, finding a trip log with specific characteristics, for example a high lateral acceleration in a long list of log files is is difficult. Therefor a method is developed for presenting key numbers of the available data in a way that gives more information about the characteristics of a trip at a glance. The MATLAB function *TripCharesteristics* automatically generates a table containing all trips logged and presents key figures for these logs for easy recognition of trips. An interactive map is generated along side the table, where the route driven is plotted when a trip is selected in the table. Figure 3.4, shows what this application looks like. A color system is integrated in the plotted route to link the key figures displayed in the table to a location. If a certain key figure is selected in the table for a trip, for example max lateral acceleration, the plotted route will show the magnitude of the lateral acceleration categorized in three level low, medium and high displayed as the colors green, yellow and red respectively. The function *TripCharesteristics* requires a fingerprint MATLAB table as an input so, the function does not have to load all trip logs one by one every time the function is used to determine the key figures.

3.4.1 Creating the fingerprint table

The fingerprint table required for the trip charesteristics tool can be created with the MATLAB function *create_fingerprint*. The creates a table with key metrics of trips for the years specified by the user. For the key figures the following metrics are selected:

- Trip code, based on the trips file name containing the trip classification date and time
- Total distance driven
- Total energy consumed
- Average energy consumed
- Max velocity
- Average velocity
- Max longitudinal acceleration
- Max lateral acceleration



TripCode	Totaldistance	Max_ax	Max_ay	Energyconsumed	maxv
VMScty_20130226_1625…	12.9979	3.8402	2.9091	1.9160	
VMScty_20130422_1338…	28.9708	3.6431	3.1574	3.8900	
VMScty_20130506_112908	228.0755	2.5617	3.5902	22.9940	
VMScty_20130528_081130	113.6869	2.5712	3.0395	13.8710	
VMScty_20130530_115041	113.7997	2.9976	3.2576	11.9770	
VMScty_20130604_1412…	20.9162	2.7222	3.6178	2.9530	
VMScty_20130604_1548…	6.0673	3.2286	3.7538	0.9940	
VMScty_20130605_1248…	47.4899	6.7406	3.4675	6.1380	
VMScty_20130606_1324…	49.3829	3.1427	4.1968	6.3570	
VMScty_20130626_0919…	12.8799	2.4574	3.3462	1.8490	
VMScty_20130626_0946…	70.2918	1.7430	2.0269	6.3980	
VMScty_20130626_111147	10.5491	2.9811	3.6620	1.5780	
VMScty_20130712_0940…	144.1527	3.7116	3.7779	15.5820	
VMScty_20130716_0903…	37.8588	2.3844	1.6564	3.9370	
VMScty_20130716_0940…	1.6957	1.7380	1.1825	0.1490	
VMScty_20130716_0950…	61.1047	1.2220	2.1229	4.6800	
VMScty_20130716_112705	16.3252	3.8981	3.2612	1.9410	
VMScty_20130930_0734…	8.1839	2.7522	3.3892	1.2100	
VMShgw_20130226_152…	12.8885	2.8906	2.5131	1.9430	
VMShgw_20130501_125…	133.6498	3.5648	3.4401	18.3590	
VMShgw_20130516_151…	59.1746	3.4099	3.5733	8.4200	
VMShgw_20130521_093…	113.6948	2.3833	2.9844	14.9830	
VMShgw_20130522_100…	113.6490	2.8764	3.2819	17.6600	
VMShgw_20130617_191…	113.4149	2.6994	2.9096	18.7190	
VMShgw_20131211_131…	55.4161	3.0021	3.8555	9.1210	
VMShgw_20131216_135…	35.3685	4.1046	4.6812	5.8820	
VMScmp_20140115_133…	0.0068	0.8914	0.4571	0.0360	
VMScmp_20140131_100…	1.3469	2.0855	2.6234	0.2740	
VMScmp_20140206_124…	0	0.0523	0.0065	0.0010	
VMScmp_20140206_134…	0	-0.0219	0.0121	0	
VMScmp_20140206_140…	0	0.0086	0.0219	0.0010	
VMScmp_20140206_144…	1.6593	2.8896	3.3116	0.3600	

Figure 3.4: Impression Trip overview Tool

Chapter 4

Vehicle dynamics analysis

4.1 Wheelspeeds

The Lupo 3L steering system is shown in figure 4.1. This steering system is a rack pinion type system. The steering ratio i_s, is the ratio between the rotation of the steering wheel δ_{sw} and the steering angle of the front wheels δ_w.

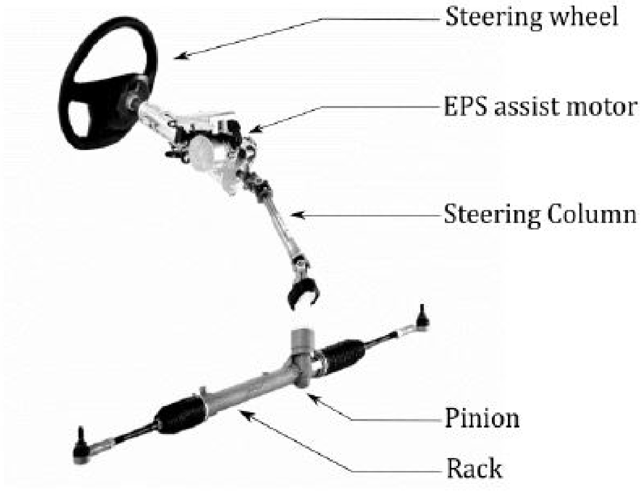


Figure 4.1: Steering system Lupo 3L

This steering ratio i_s is was validated by experiment performed in report [4]. In this experiment, the wheel angles were measured by placing the front wheels on two frictionless plates, which are able to measure the angle. The signal for the rotation of the steering wheel was logged by the VMS-CAN data acquisition system, this signal may include a certain offset with respect to the actual steering wheel position. Therefore it was also manually noted when the steering wheel was at a position of 90; 180 or 270 degrees and compared to the measured, to find the offset. The measured angles at the left and right wheel are plotted as a function of the steering wheel angle, shown in Figure 4.2. This figure shows that relation is linear for small steering wheel angles and shows different behaviour for larger angles. This behaviour for larger angles is a result of the Ackermann steering geometry. With Ackermann steering geometry the inner wheel has a larger steering angle than the outer wheel. When a vehicle is making a turn, the inside wheel must follow a tighter curve than the outside wheel. To achieve this, the geometry of the steering system must be arranged to turn the inside wheel with a larger angle than the outside wheel.

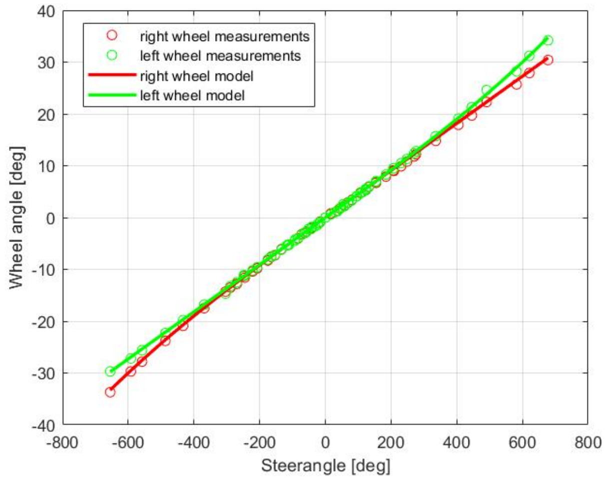


Figure 4.2: Steer angle as a function of the steering wheel angle

The Ackermann steering geometry, is validated by deriving a model for the different wheelspeeds with as input a velocity trajectory and steering wheel inputs and comparing this to measured wheelspeeds. Figure 4.3 shows kinematic model used.

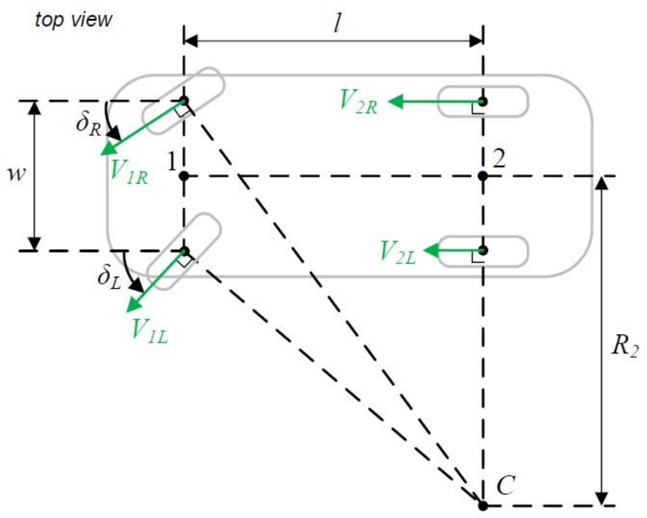


Figure 4.3: Ackerman steering angles of the front wheels of a passenger car

For an object traveling in a circular motion its speed v can be calculated by multiplying the vehicle's angular velocity ω_z around the centre of a corner, point C in Figure 4.3, with the radius of the corner R_2:

$$v = \omega_z R_2 \tag{4.1}$$

Using this relation the angular velocity ω_z of the vehicle can be calculated using the steerangle and the longitudinal velocity. First the distance from rotation point C to the centre line of the car can be found with trigonometric functions:

$$R_2 = \frac{l}{\tan \delta} \tag{4.2}$$

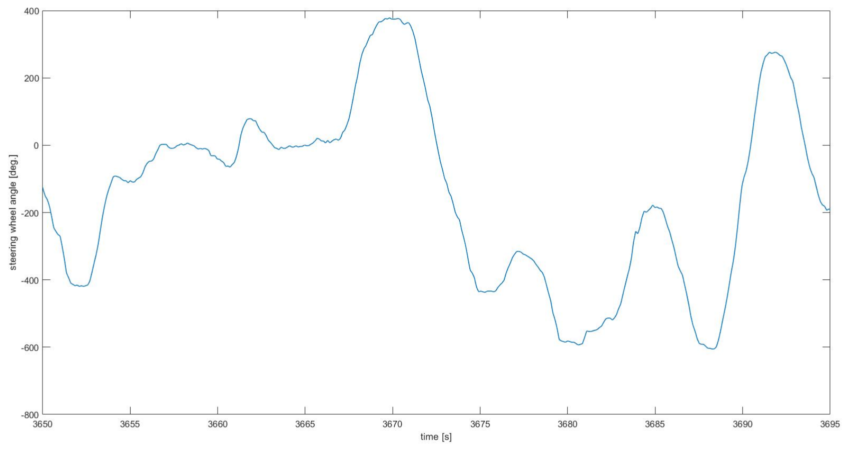
Where l equals the wheelbase of the vehicle and δ the steerangle of the centre line of the vehicle which equals $\frac{\delta_{sw}}{i_s}$. Then according to (4.1) it holds that:

$$\omega_z = \frac{v}{R_2} \tag{4.3}$$

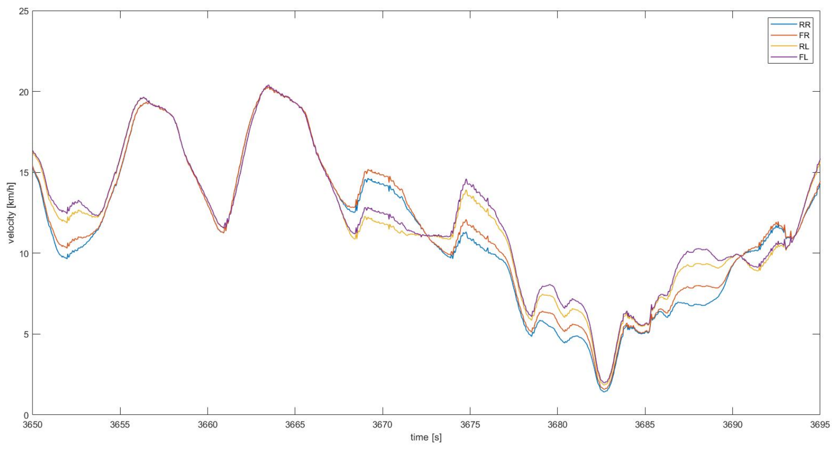
Using this angular velocity and trigonometric functions to determine the distance from point C the following models for each wheel speed was derived:

$$\begin{aligned} v_{2l} &= (R_2 - \tfrac{w}{2}) * \omega_z \\ v_{2R} &= (R_2 + \tfrac{w}{2}) * \omega_z \\ v_{1L} &= \sqrt{(R_2 - \tfrac{w}{2})^2 + l^2} * \omega_z \\ v_{1R} &= \sqrt{(R_2 - \tfrac{w}{2})^2 + l^2} * \omega_z \end{aligned} \tag{4.4}$$

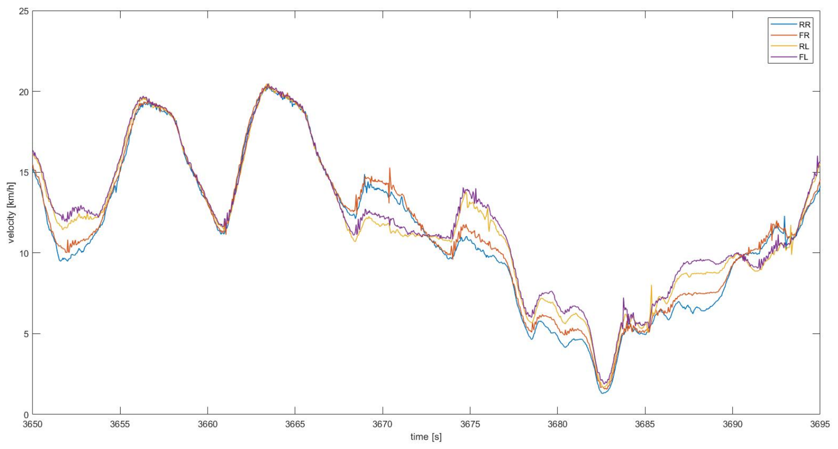
This model is evaluated by plotting recorded steerangles for trip "VMScty_20130604 _141201", between t= 3650 and t= 3695 [s] as well as the corresponding measured wheelspeeds and modeled wheelspeeds. The plots are shown in Figure 4.4. As can be seen from this figure the model tracks the measured wheelspeeds quite well confirming Ackermann steer geometry. The error of the model is quantified by taking the mean of the absolute difference between the measured wheel speed and modeled wheel speed for each wheel. For the front left wheel the error is 0.11 [km/h], for the front right wheel 0.13 [km/h], for the rear left wheel 0.12 [km/h] and for the rear right wheel 0.11 [km/h].



(a) input steerangle



(b) modeled wheelspeeds



(c) Measured wheel speeds

Figure 4.4: Comparison Ackermann steering model and measured data

4.2 Single track vehicle model

The single track vehicle model is physically plausible representation of the cornering behaviour of vehicles without, major modeling and parameterization effort. The cornering behavior of the Lupo EL will be analyzed using this model, as depicted in figure 4.5.

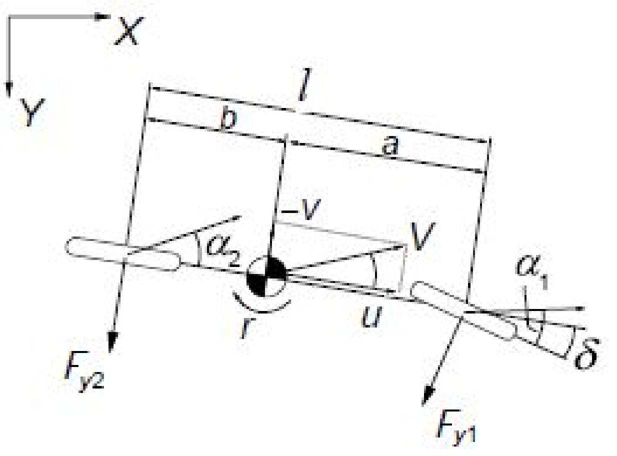


Figure 4.5: Single Track vehicle model

For steady-state cornering given a corner radius R and vehicle speed V the required steering angle is given by relation (4.5), derivation of this relation can be found in [2].

$$\delta = \frac{l}{R} - \frac{mV^2}{Rl}(\frac{a}{C_2} - \frac{b}{C_1}) \tag{4.5}$$

Where l equals the wheelbase of the vehicle and R the corner radius. Furthermore, m equals the mass of the vehicle, V velocity and $C_{1,2}$ the cornering stiffness. this relation has two contributions:

- $\frac{l}{R}$, "kinematic" part (Ackermann steer)
- $\frac{mV^2}{Rl}(\frac{a}{C_2} - \frac{b}{C_1})$, speed (or lateral acceleration) dependent part

Expression (4.5) can be simplified by introducing the understeer coefficient or understeer gradient η:

$$\eta = \frac{mg}{l}(\frac{b}{C_1} - \frac{a}{C_2}) \tag{4.6}$$

and using the the relation between the lateral acceleration a_y, velocity V and cornering radius R:

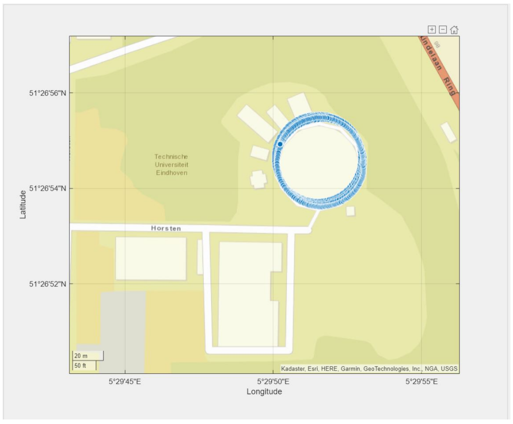
$$a_y = \frac{V^2}{R} \tag{4.7}$$

The steerangle for a given corner radius and lateral acceleration is given by:

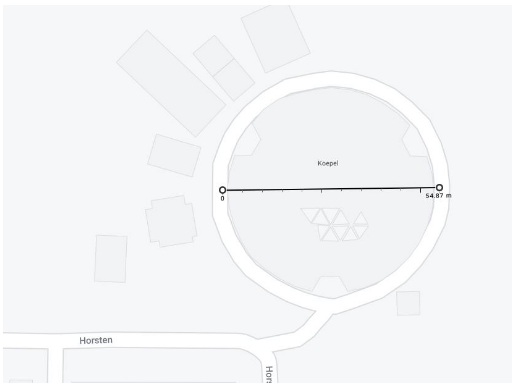
$$\delta = \frac{l}{R} + \frac{a_y}{g}\eta \tag{4.8}$$

4.2.1 Determination of the understeer coefficient

In (4.6), the understeer coefficient is introduced. This coeffcient is dependant on certain vehicle parameters such as mass, wheelbase and cornering stiffness, of which the cornering stiffness is unknown. The understeer coefficient can be determined by plotting the steering angle as a function of the lateral acceleration for a constant corner radius. Using the fingerprint tool, a trip was selected with a circular trajectory, where the corner radius R is constant. The trip 'VMScmp_20141124_124722', shown in Figure 4.10a, is used to determine the understeer coefficient. Using Google Maps the Diameter of the circular trajectory between the coordinates (51.44847 5.49726) and (51.44847 5.49805) was determined to be 54.87 m, see Figure 4.10b.


Latitude
51°26'56"N
51°26'54"N
51°26'52"N
Technische Universiteit Eindhoven
Horsten
20 m
50 ft
Kadaster, Esri, HERE, Garmin, GeoTechnologies, Inc., NGA, USGS
5°29'45"E
5°29'50"E
5°29'55"E
Longitude

(a) trip 'VMScmp_20141124_124722'



(b) Trajectory diameter Google Maps

The signal m.controls.steersangle of the trip-log was plotted as a function of the signal m.chassis.ay, this is shown in Figure 4.7. The signal m.controls.steerangle was first converted to the steer angle of the front wheels using the steering ratio i_s, as described in subsection 4.1. Furthermore, only data points for which the longitudinal acceleration is small (between -0.5 $[\frac{m}{s^2}]$ and 0.5 $[\frac{m}{s^2}]$) and for which the speed average speed of the real wheels is above 7 $[\frac{m}{s}]$ are included. Along with the measured data the line $\frac{l}{R}$ is plotted in red and the linear regression model, found by the backslash operator in MATLAB between the data points, is plotted in yellow. The backslash operator in MATLAB uses a least-squares linear regression fit to derive a data model for supplied data. The linear regression model has a slope of 0.0059 and an initial value almost exactly the same as the value of $\frac{l}{R}$. Therefore, it can be concluded that term $\frac{\eta}{g}$ is equal to 0.0059 and that the understeer coefficient η is 0.058 [rad]. A positive value for the understeer coefficient η means that the Lupo EL is a understeered vehicle meaning that to maintain a constant radius R while increasing forward speed V, the steering angle must increase.

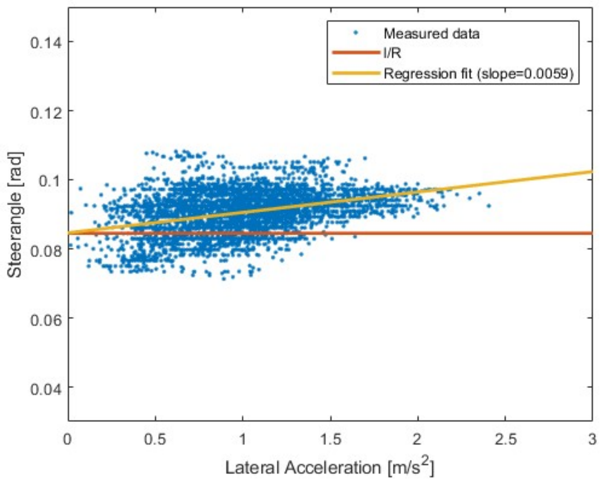


Figure 4.7: Understeer coefficient η as determined from drivecycle'VMScmp_20141124_124722'

4.2.2 Validation single track model

The single track vehicle model, with an understeer coefficient of 0.058[rad], is subjected to vehicle data recorded throughout the years. The single track model 4.8 assumes that the vehicle is driven in a circle with fixed radius R. The data recorded contains a lot of trips with varying, corners. So, simply plotting the steerangle as a function of lateral acceleration for the recorded data wont give the desired correlation as there is no fixed radius. To compensate for the fixed radius constraint the following relations can be used:

$$R = \frac{V}{z} \approx \frac{u}{\omega_z} \tag{4.9}$$

and

$$\frac{l}{R} = \frac{l\omega_z}{u} \tag{4.10}$$

These relations allow the single track vehicle model to be rewritten into the form independent of the Corner radius R:

$$\delta - \frac{\omega_z}{u} l = \frac{\eta}{g} a_y \tag{4.11}$$

Figure 4.8, shows how the single track model with an understeer coefficient of 0.058[rad] compares to the measured data. The measured data only considers the data point with the following conditions:

- longitudinal accelerations between -0.5$[\frac{m}{s^2}]$ and 0.5$[\frac{m}{s^2}]$.
- average wheel speed of the real wheels above 7.5 $[\frac{m}{s}]$

For a more clear comparison the data was divided in sections corresponding to a stepsize in lateral acceleration of 1 $[\frac{m}{s^2}]$ then for each section a linear regression model was made using the backslash operator in Matlab. The signals used for the linear regression models are displayed in Table 4.1

Signal name	Unit
m.controls.steerangle	deg
m.chassis.wheelspeed_RL	km/h
m.chassis.wheelspeed_RR	km/h
m.chasssis.yawrate	deg/s

Table 4.1: Logged signals used for comparison single track model

The signal m.controls.steerangle was first converted to the steerangle of the front wheels using steering ratio i_s=22. For the lateral acceleration the average speed of the rear wheels and the yawrate are used as the sensors used for these signals contain less noise than the lateral acceleration sensor (m.chassis.ay). The y-intercept points and the slopes of the linear regression models are used to represent the data in Figure 4.8, see Appendix C for the full data set. The data for the left-hand turn is mirrored for easy comparison with the right turn.

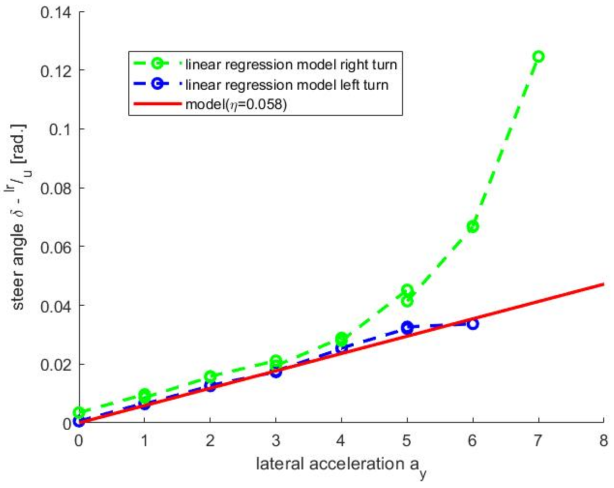


Figure 4.8: Comparisson single track model and linear regression fits measured data

As can be seen, the regression data models confirm that the Lupo EL is an understeered vehicle as the slope is positive, indicating that the understeer coefficient must be positive. Furthermore, the single track model correlates quite well to the data regression models up to a lateral acceleration of 4 $[\frac{m}{s^2}]$. After that point the regression models have a different slope meaning that there is non-linear behaviour for lateral accelerations above 4 $[\frac{m}{s^2}]$.

4.2.3 Yaw velocity gain

Yaw velocity gain can also be used to validate understeer behaviour without the need for the vehicle data to be measured for a fixed corner radius R. The yaw velocity gain can be determined in the following way:

$$\frac{r}{\delta} = \frac{V/l}{1 + \frac{\eta}{gl}V^2} \tag{4.12}$$

Plotting this equation with understeer coefficient $\eta = 0.058$ over the measured data, where again linear regreassion models have been taken for the data per velocity intervals of 3 $[\frac{m}{s}]$, the full data set can be found in Appendix C. Figure 4.9, shows the plot of the model along with the linear regression fit for right and left turn data and the yaw velocity gain of a neutral steered vehicle.

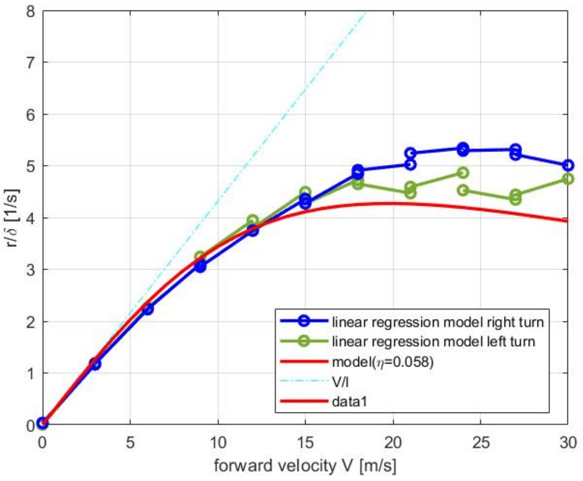
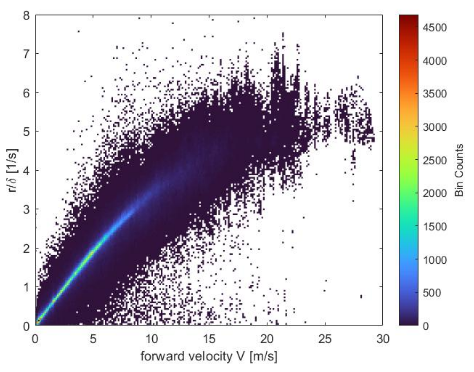
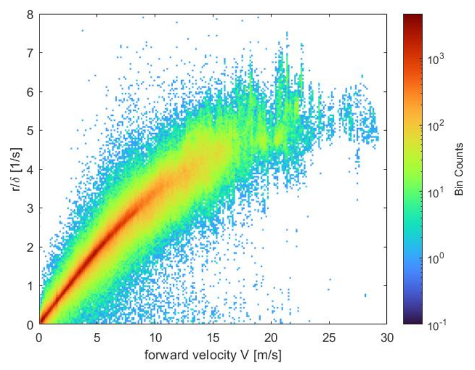


Figure 4.9: Comparison single track model and linear regression fits measured data yaw velocity

Figure 4.9 shows that the measurements and the model agree well up until 15 $[\frac{m}{s}]$. After that the deviation between the model and measurements get significantly larger. Also especially for the measurements of the left turn after 15 $[\frac{m}{s}]$ the linear regression models don't overlap well and the slope of the models changes significantly so, there is no clear correlation in the data there. This can probably be attributed to the fact that there just is not much data recorded for velocities higher than 15 $[\frac{m}{s}]$. This is demonstrated by binscatterplots presented in figure??. As can be seen the density of of data points after 15 $[\frac{m}{s}]$ is significantly lower than below 15 $[\frac{m}{s}]$.



(a) binned scatterplot yaw velocity gain data



(b) binned scatterplot yaw velocity gain data log color scale

Chapter 5

Conclusions & recommendations

In this chapter, the project goals from subsection 1.1 are revisited and the study results will be evaluated with the use of these goals.

5.1 Conclusions

The first goal of this project was to streamline the processing of the CAN logger measurement files to a usable format in MATLAB. It had to made robust to cover exceptions (e.g. missing GPS data) and able to handle all available measurement files. This goal is met with the development of a conversion script in MATLAB. This script converts all measurement files logged by the VMS-CAN data acquisition system into well structured MATLAB files. All signals present in these files have been resampled to the same sample frequency. Missing GPS data in VMS measurement files has been accounted for by adding GPS data acquired by the built-in tablet using the "Torque" application. SRTM elevation data has been added to the final MATLAB measurement files to compensate for inaccuracies in elevation data acquired by both the GPS sensor of the vehicle aswell as from the tablet.
The goal to develop a method/metric to characterize a trip is achieved by comparing data from three distinct driving scenario's: Urban, Highway, Campus. Based on that comparison it was concluded that the best way to characterize a trip is by first evaluation the location data to determine if the trip was on the TU/e campus. The distance of the trip and the average velocity are analyzed to make a distinction between urban and highway trips. To make it more easy to identify what data is available within a certain measurement file a fingerprint tool was developed that displays certain key numbers of a trip along with an interactive map for visualization of the trip. This tool is described in detail in this report.
Finally, to analyze vehicle dynamics a model was derived that models the wheelspeed for all four wheel separately based on Ackermann steering geometry. The model takes a velocity profile and steering wheel angles as input. The model was validated using measurement data and confirmed that the Lupo EL has Ackermann steering geometry. A single track vehicle model was derived to analyze cornering behaviour. This single track vehicle model was compared to measurement data. From the comparison of the single track vehicle model and measured steerangle data plotted as a function of lateral acceleration it was concluded that the model is correlates well to the data up to a lateral acceleration of 4 [m/s^2]. The understeer behaviour of the vehicle was also analyzed using a model for yaw velocity gain, this model was validated with measurement data. From this validation it was concluded that the model correlates well with measured data up to a forward velocity of 25[m/s] above that non-linear behaviour occurs in the measurement data.

5.2 recommendations

- In chapter 3.2 metrics are defined to categorize driving scenario's of a trip. However there still is no information of the different road types that was driven on during one trip. A method is developed to use OpenStreetMap to retrieve this information. The method developed was concluded not reliable, so future research needs to be done to further improve the method before it can be used.
- In chapter 4.2 a single track vehicle model is derived to analyse the cornering behaviour of the vehicle. This model is evaluated using available measurement data. It was concluded that the model does not take non-linear behaviour into account for high lateral accelerations and high forward velocities. Therefore the model needs to be improved accounting for this non-linear behaviour.

Bibliography

[1] Camiel J.J. Beckers, Mário Paroha, Igo J.M. Besselink, and Henk Nijmeijer. A microscopic energy consumption prediction tool for fully electric delivery vans. In *33rd World Electric Vehicle Symposium Exposition (EVS33) Peer Reviewed Conference Papers*. Electric Vehicle Symposium and Exhibition, September 2020. 33rd International Electric Vehicle Symposium and Exposition (EVS33), EVS33 ; Conference date: 14-06-2020 Through 17-06-2020. 8, 12

[2] Dr. Ir. I.J.M. Besselink. Tue_4AT000_2020.pdf. Based on: Tire and Vehicle Dynamics by Hans Pacejka Third edition 2012, Butterworth-Heinemann, ISBN 978-0-08-097016-5. 19

[3] I.J.M. Besselink, P.F. Oorschot, van, and H. Nijmeijer. Design of an efficient, low weight battery electric vehicle based on a vw lupo 3l. In *25th International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium (EVS 25), 5-9 November 2010 Shenzen, China*, pages 32–41, 2010. 2

[4] M.H.M. Merks. VW Lupo EL: Data analysis. 5(1):14–23, 2015. DC2015.0947. 15

[5] Inc. The MathWorks. finddelay. urldate: 2022-07-07. 6

[6] Paul F. van Oorschot, Igo J.M. Besselink, Erwin Meinders, and Henk Nijmeijer. Realization and control of the lupo el electric vehicle. *World Electric Vehicle Journal*, 5(1):14–23, January 2012. 2

Appendix A

Overview message ID's CAN systems

CH	ID	Active	Message Description	Converted?	Converted_name
1	80	no	Receive_Airbag		
1	112	yes	Cruise_Control	yes	controls.CC.p,controls.CC.I,controls.CC.D
1	113	yes	Inverter_inputs	yes	motor.inverter.PWM
1	114	yes	Cruise_Control_2	yes	controls.CC.CruiseSpeedDown, Speedup,Speed,PID_Output
1	128	yes	OPD_1		
1	129	yes	OPD_2	yes	controls.OPD.brakelight,controls.OPD.tau
1	144	yes	Transmit_error_gear	yes	controls.gearlever
1	273	yes	Transmit_Chargplug	yes	temp.chargeplug
1	277	yes	Transmit_HV1	yes	HV.voltage/current/power/DCDC
1	278	yes	Transmit_HV_2	yes	HV.nrg_discharge/nrg_regen
1	288	yes	Transmit_DCDC_1	yes	LV.current.DCDC
1	289	no	Transmit_DCDC_2		
1	304	yes	ESP_YAW_lat_acc	yes	chassis.ay
1	305	yes	ESP_Long_acc	yes	chassis.ax,chassis.yawrate
1	321	no	Transmit_Isometer		
1	337	yes	Transmit_Lupo_1	yes	controls.steerangle
1	338	yes	Transmit_Lupo_2	yes	temp.coolant
1	352	yes	Transmit_LV_1	yes	LV.current.battery/heater
1	353	yes	Transmit_LV_2		
1	368	yes	Transmit_PLC_1	yes	LV.voltage.PLC/temp.PLC
1	369	yes	Transmit_PLC_2		
1	370	yes	Transmit_PLC_3		
1	371	yes	Transmit_PLC_4		
1	384	yes	Transmit_Brake_1		controls.brake.pos/controls.brake.pres
1	385	yes	Transmit_Brake_2	yes	controls.throttle.pos,controls.accelerator.rawpos
1	386	yes	Transmit_Brake_3		
1	387	yes	Transmit_Brake_4	yes	controls.throttle.released, controls.throttle.kickdown,controls.brake.switch
1	388	yes	Transmit_Brake_5		
1	400	yes	Transmit_Power_1	yes	LV.current/fan/pump/dtrlights/tablet
1	401	yes	Transmit_Power_2	yes	LV.current.gearlever/inverter/charger
1	402	yes	Transmit_Power_3	yes	LV.current.BMX/vacpump

CH	ID	Active	Message Description	Converted?	Converted_name
1	403	yes	Transmit_Power_4		
1	416	no	Receive_ABS_1		
1	640	no	ECU280H		
1	648	no	ECU288H		
1	800	no	Receive_Dash_1		
1	896	no	ECU380H		
1	904	no	ECU388H		
1	1056	no	Receive_Dash_2		
1	1088	no	TCM440H		
1	1096	no	Receive_Poke		
1	1152	no	ECU480H		
1	1160	no	ECU488H		
1	1184	no	Receive_ABS_2	yes	chassis.whellspeed_FL,FR,RL,RR,chassis.s
1	1312	yes	Receive_Dash_3		
1	1344	yes	TCM540H	yes	controls.ecomode
1	1352	no	TCM548H		
1	1408	no	ECU580H		
1	1440	yes	Receive_ABS_3		
1	1488	no	Receive_Dash_4		
1	1496	no	Receive_Dash_5		
1	419358425	no	GPS_Time_Date		
1	419358937	no	GPS_Direction_Speed		
1	419361753	no	GPS_Position		
1	419412441	no	GPS_Status		
2	1568	yes	Recieve_BMS_1		
2	1569	yes	Recieve_BMS_2		
2	1570	yes	Recieve_BMS_3		
2	1571	yes	Recieve_BMS_4	yes	HV.cell_Vmi/Vmin_nr/Vmax/Vmax_nrn
2	1572	yes	Recieve_BMS_5		
2	1573	yes	Recieve_BMS_6		
2	1574	yes	Recieve_BMS_7	yes	HV.soc/DOD
2	1575	yes	Recieve_BMS_8	yes	temp.cell_low/low_nr/high/high_nr
2	1576	yes	Recieve_BMS_9		
2	1552	yes	Receive_Charger_status		
2	1553	yes	Receive_Charger_1	yes	charger.DC_voltage,DC_current,DC_energy, AC_voltage,AC_current/AC,$_{e}nergy$
2	1554	yes	Receive_Charger_2		
2	1555	yes	Receive_Charger_temp	yes	temp.charger/pack1/pack2/pack3
2	1556	yes	Receive_Charger_error		
2	301	yes	Receive_Inverter_data	yes	motor.rpm/torque
2	1607	yes	Receive_Inverter_status	yes	controls.regenbraking
2	1577	yes	Receive_BMS_10		

Table A.1: Overview Message ID's CAN bus

Appendix B

Data conversion script

```
clear
clc
%% Add paths for VMS Matlab functions and SRTM elvation data files

addpath(genpath('VMS_MatlabTools\'))
%Add folder with other functions
addpath('./VMS_MatlabTools/getSRTMelevation/Functions');
%Add Readhgt by Francois Beauducel:
addpath('./VMS_MatlabTools/getSRTMelevation/Functions/Readhgt')
%Add FilterM by Jan Simon (replaces signalprocessing toolbox)
addpath('./VMS_MatlabTools/getSRTMelevation/Functions/FilterM');

%Define folder where SRTM maps are stored:
folders.SRTM = './VMS_MatlabTools/getSRTMelevation/Data/SRTM';

makePlots= false;

%% Set locations where data should be loaded from and saved to


% Set location where converted files should be stored (Either by pasting the ...
    location manually or by Matlab ui, command out the not desired option)

SelectLocDataStore = 'C:\Users\matth\OneDrive - TU Eindhoven\Documents\TUe\...
    BEP2\Data';
% SelectLocDataStore = uigetdir("C:\",'select folder where the data should be...
     stored');

%Select bin files that will be conveted to matlab structures
[SelectedBinFiles, pathSelectedBinfiles] = uigetfile('*.bin','Select one or ...
    More BIN Files that should be converted','MultiSelect','on');

%Set Location where Torque logs are saved (Either by pasting the location ...
    manually or by Matlab ui, command out the not desired option)

SelectLocTorquefiles = 'C:\Users\matth\OneDrive - TU Eindhoven\Documents\TUe\...
    BEP\TorqueToMat\2015';
% SelectLocTorquefiles = uigetdir("C:\",'select folder where the Torque files...
     are stored');

if isequal(class(SelectedBinFiles),'char')
    SelectedBinFiles=cellstr(SelectedBinFiles);
end
LocBinfiles = strcat(pathSelectedBinfiles,SelectedBinFiles);
addpath(pathSelectedBinfiles);

%% Initialize error detection
```

```matlab
 %creates variable to track wheter VMSconv Functions is run succesfull
VMSconvFail=0;

%creates variable to track wheter VMS2Mat Functions is run succesfull
VMS2MATFail=0;

%remove scratchdir to prevent errors with VMS2MAT
 if isfolder('scratchDir')
    rmdir("scratchDir\","s");
end



%% Convert Bin files to .mat and resample

%Loop over all Bin files
for i=1:length(LocBinfiles)

     %Open txt file to log errors
     fileID = fopen('Errors.txt','w');

     copyfile(LocBinfiles{i},pwd);
     copiedFile = strcat(pwd,'\',SelectedBinFiles{i});

     % Setup VMS2MAT.exe VMS-tool by venamics to load data bin files into ...
         MATLAB
     arg0 = strcat('"VMS2MAT.exe"',32,'"',copiedFile,'"');

     % Try VMS2MAT.exe VMS-tool by venamics to load data bin files into
     % MATLAB, catch if an error occurs
     try
         system(arg0);
     catch ME
         %Check if scratchDir folder is present and remove, as the
         %scratchDir folder often gives erros
            if isfolder('scratchDir')
               rmdir("scratchDir\","s");
               % Try again wiht scratchDir removed
               try
                    system(arg0)
               catch ME2
                    % If still fails move bin file to corrupted Bin files
                    % folder and write error in the error.txt file
                    movefile(CopiedFile,"Corrupted Files\Bin\");
                      fprintf(fileID, strcat('Bin file',copiedFile,'Failed to ...
                          convert using VMS2MAT.exe , error:',ME2.message,'||','...
                          \n'));

                      VMS2MATFail=1;
                      clear ME2
               end

            else
                % If there is no scratchfir folderpresent, move bin file to ...
                     corrupted Bin files
                % folder and write error in the error.txt file
               movefile(CopiedFile,"Corrupted Files\Bin\");
               fprintf(fileID, strcat('Bin file',copiedFile,'Failed to convert ...
                   using VMS2MAT.exe, error:',ME.message,'||','\n'));

               VMS2MATFail=1;
               clear ME
            end

     end

%check again wheter a scratchDir folder is left behind and remove if
```

```
%necessary
     if isfolder('scratchDir')
         try
         rmdir("scratchDir\","s");
         catch
         end

     end

     %check wheter a .mat file was created by VMS2MAT.exe, if not set
     %Failbool to 1 and move bin file that should have been converted to
     %Corrupted Bin files folder
     d = dir('VMSlog∗.mat');
     if isempty(d)
         VMS2MATFail=1;
         movefile(copiedFile,"Corrupted Files\Bin\");

     else
         %If the copied file is succesfully converted, the copied Bin file
         %can be removed for this directory
         delete(copiedFile);
     end

     % If the conversion of the bin file to Matlab was succesfull it the
     % data will be resampled using the function VMSconvMATv2
     if VMS2MATFail≠1
        try
            VMSconvMATv2(d.name);
            % If VMSconvMATv2 results in an error the error will be logged
            % in the error.txt file and the .mat file will be moved to the
            % corrupted MAT files for manual check
        catch ME3
            movefile(d.name,"Corrupted Files\Mat\");
            fprintf(fileID, strcat('Mat file',d.name,'Failed to resample using...
                 VMSconvMATv2, error:',ME3.message,'','\n'));

            VMSconvFail=1;
            clear ME3, clear d
        end

     end

     if VMSconvFail ≠ 1 && VMS2MATFail ≠1
         delete(d.name)
         clear d
         d =[dir('VMSchg∗.mat');dir('VMScty∗.mat');dir('VMShgw∗.mat');dir('...
             VMScmp∗.mat')];
           if ¬isfolder(strcat(SelectLocDataStore,'\',d.name(8:11)))
              mkdir(strcat(SelectLocDataStore,'\',d.name(8:11)));
         end

%% Move charge to its own folder




        %% Find and add Torque data
        load(d.name)
     if  d.name(4:6) ≠ convertCharsToStrings('chg') && ¬isfield(m,'gps')
      dateVMSfile = d.name(8:15);
      dateVMSfile = datestr(datetime(dateVMSfile,'InputFormat','yyyyMMdd'));
      dateVMSfile = strcat(dateVMSfile(8:11),dateVMSfile(3:7),dateVMSfile(1:2)...
          );
      timeVMSfile = d.name(17:22);

      Torquedir = dir(fullfile(SelectLocTorquefiles,strcat('∗',dateVMSfile,'∗'...
          )));
```

```

    if isempty(Torquedir)
        dateVMSfile=append(dateVMSfile(1:8),'.',dateVMSfile(9:11));
        Torquedir = dir(fullfile(SelectLocTorquefiles,strcat('*',dateVMSfile...
            ,'*')));
    end


    if size(Torquedir,1) == 1
        load(d.name);
        copyfile(fullfile(Torquedir.folder,Torquedir.name));
        try
        [TorqueData,t12,ME4]=TorqueconvMAT(Torquedir.name,d.name,folders,...
            makePlots);
        catch
            TorqueData='No torque Match found';
            t12=0;
        end
       m.TorqueData=TorqueData;
        save(d.name,'m');
    elseif size (Torquedir,1) > 1
        load(d.name);
        disp('many trips that day')
        for j=1:size(Torquedir,1)
            copyfile(fullfile(Torquedir(j).folder,Torquedir(j).name));
            try
            [¬,t12(j),¬] = TorqueconvMAT(Torquedir(j).name,d.name,folders,...
                makePlots);
            catch
                t12(j)=0;
            end
        end
        idx = find(t12 ≠ 0);
        if isempty(idx) == 0
            try
           [TorqueData,t12,ME4]=TorqueconvMAT(Torquedir(idx).name,d.name,...
               folders,makePlots);
            catch
                TorqueData='Multiple Torque matches found check by hand ';
            end
           m.TorqueData=TorqueData;
           save(d.name,'m');
        else
           m.TorqueData ='No torque Match found';
           save(d.name,'m');
        end

    elseif isempty(Torquedir)

        load(d.name);
       m.TorqueData ='No torque Match found';
        save(d.name,'m');

    end

    if class(m.TorqueData) == "struct"
       TorqueCheck = Matchcheck(m);
    else
        TorqueCheck = 'NoMatch';


    end


  if (TorqueCheck == "WrongMatch" || class(m.TorqueData) ≠ "struct") && ¬...
       isempty(Torquedir)
       Torquedata=rematch(Torquedir,d.name,folders,makePlots);
```

```
          m.TorqueData=Torquedata;
           save(d.name,'m');
    end

    if class(m.TorqueData) == "struct"
    TorqueCheck = Matchcheck(m);
    end

    if TorqueCheck== "GoodMatch"
    movefile(d.name,strcat(SelectLocDataStore,'\',d.name(8:11)),'f');
    end

    if TorqueCheck== "WrongMatch"
        if ¬isfolder(strcat(SelectLocDataStore,'\',d.name(8:11),'\','...
            wrongmatch'))
            mkdir(strcat(SelectLocDataStore,'\',d.name(8:11),'\','wrongmatch'...
                ))
        end
    movefile(d.name,strcat(SelectLocDataStore,'\',d.name(8:11),'\','...
        wrongmatch'),'f');
    end

    if TorqueCheck== "Timeshift"
        if ¬isfolder(strcat(SelectLocDataStore,'\',d.name(8:11),'\','...
            timeshift'))
            mkdir(strcat(SelectLocDataStore,'\',d.name(8:11),'\','timeshift')...
                )
        end
    movefile(d.name,strcat(SelectLocDataStore,'\',d.name(8:11),'\','timeshift...
        '),'f');
    end

    if TorqueCheck== "CheckbyHand"
        if ¬isfolder(strcat(SelectLocDataStore,'\',d.name(8:11),'\','check'))
            mkdir(strcat(SelectLocDataStore,'\',d.name(8:11),'\','check'))
        end
    movefile(d.name,strcat(SelectLocDataStore,'\',d.name(8:11),'\','check'),'...
        f');
    end

    if class(m.TorqueData) ≠ "struct"
        if ¬isfolder(strcat(SelectLocDataStore,'\',d.name(8:11),'\','noMatch'...
            ))
            mkdir(strcat(SelectLocDataStore,'\',d.name(8:11),'\','noMatch'))
        end
    movefile(d.name,strcat(SelectLocDataStore,'\',d.name(8:11),'\','noMatch')...
        ,'f');
    end


    elseif isfield(m,'gps')
        trk.lon = m.gps.longitude.';
        trk.lat = m.gps.latitude.';
        trk.dis = getHalversineDistance( trk.lat, trk.lon );
    try
        [trk.elev, ¬, ¬, ¬] = getSRTMelevation(trk, folders.SRTM, makePlots);
       m.gps.elevationMecpro=trk.elev;
       save(d.name,'m');
    catch
       m.gps.eletionMecpro='error using Mecpro';
    end

        movefile(d.name,strcat(SelectLocDataStore,'\',d.name(8:11)),'f');

    elseif  d.name(4:6) == convertCharsToStrings('chg')
         if ¬isfolder(strcat(SelectLocDataStore,'\',d.name(8:11),'\','Charge'...
             ))
            mkdir(strcat(SelectLocDataStore,'\',d.name(8:11),'\','Charge'))
        end
        movefile(d.name,strcat(SelectLocDataStore,'\',d.name(8:11),'\','...
            Charge'),'f');



    end




clear d
clear m
clear TorqueCheck

    end
VMSconvFail=0;
VMS2MATFail=0;
save("Errors.txt");
fclose('all');
end
```

Appendix C

Linear regression fitting of cornering data

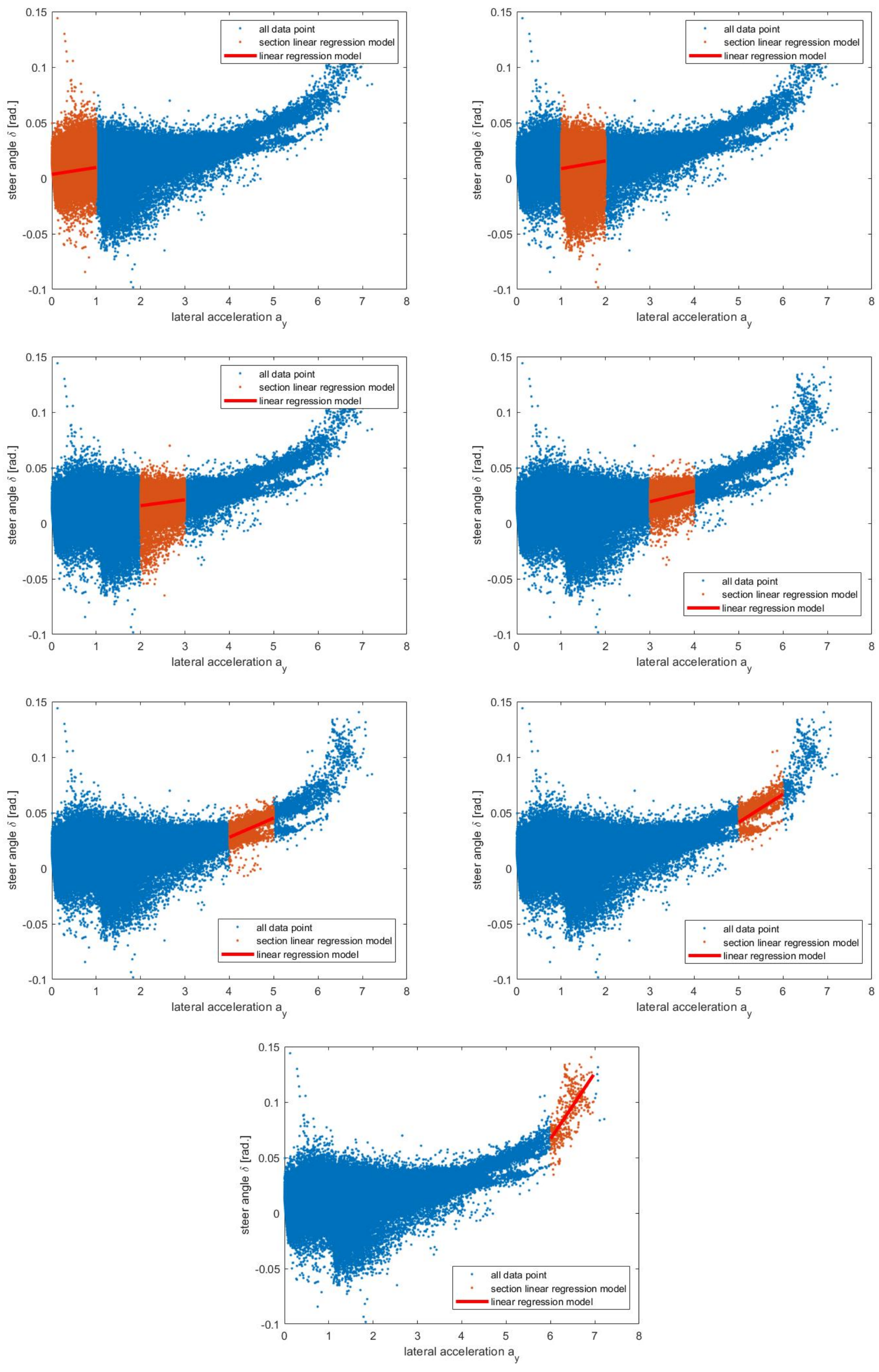


Figure C.1: Linear regression fit steerangle as a function of lateral acceleration

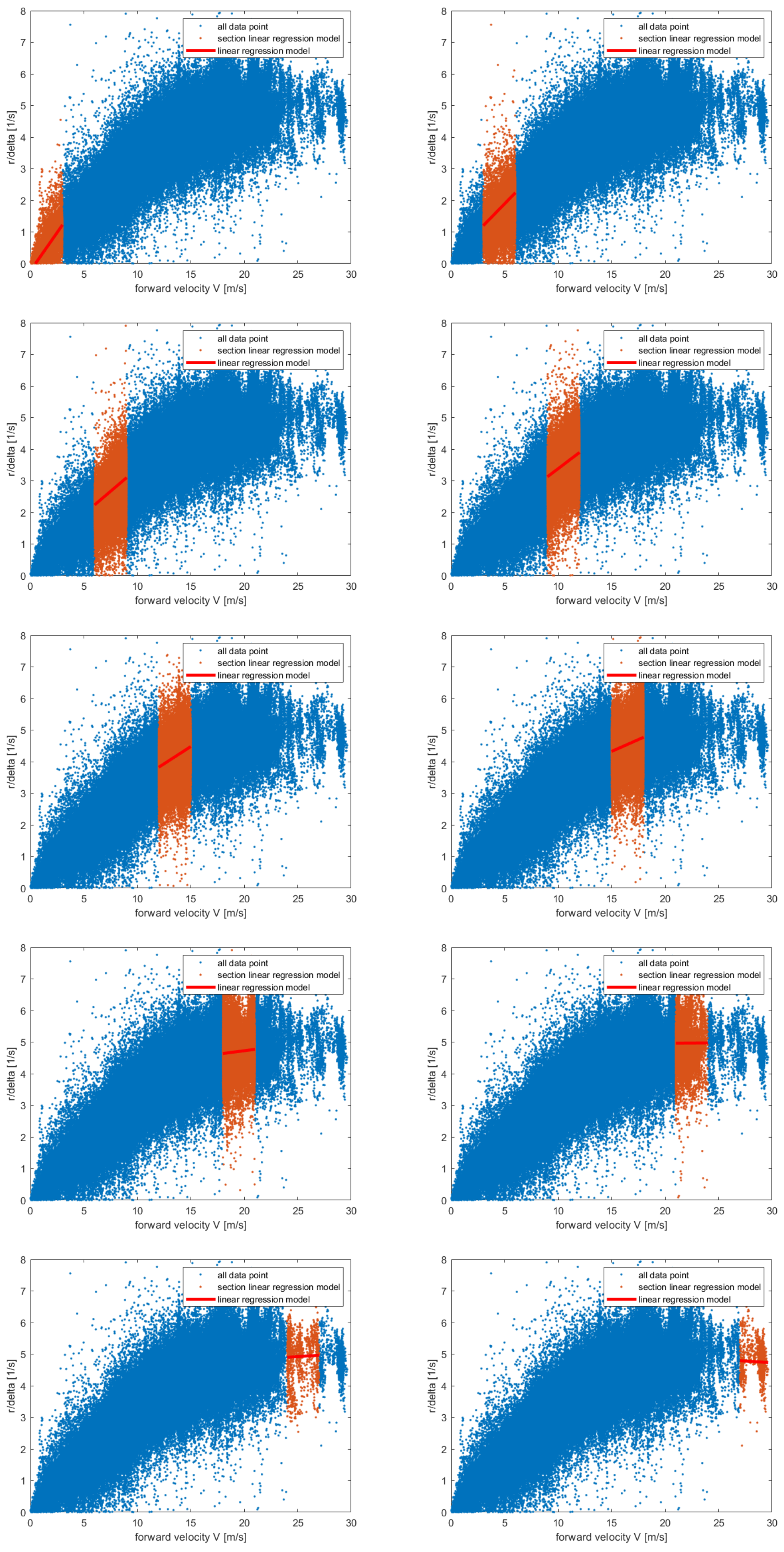


Figure C.2: Linear regression fit Yaw Velocity gain