CHALMERS
UNIVERSITY OF TECHNOLOGY

Pilot Naturalistic Riding Study (NRS) with VOI e-scooters to improve traffic safety

Project report in the course TME180 Automotive Engineering Project

DANIEL SCHMIDT

FREDRIK LINDELÖW

GOWTHAM GUNASHEKARA

RAHUL RAJENDRA PAI

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

PROJECT REPORT IN AUTOMOTIVE ENGINEERING PROJECT

PILOT NATURALISTIC RIDING STUDY (NRS) WITH VOI E-SCOOTERS TO IMPROVE TRAFFIC SAFETY

DANIEL SCHMIDT
FREDRIK LINDELÖW
GOWTHAM GUNASHEKARA
RAHUL RAJENDRA PAI

CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2022

PILOT NATURALISTIC RIDING STUDY (NRS) WITH VOI E-SCOOTERS TO IMPROVE TRAFFIC SAFETY

DANIEL SCHMIDT, FREDRIK LINDELÖW, GOWTHAM GUNASHEKARA, RAHUL RAJENDRA PAI

Supervisors: Giulio Bianchi Piccinini & Marco Dozza, Department of Mechanics and Maritime Sciences
Supervisor: André Dankert, Voi Technology AB
Examiner: Jonas Sjöblom, Department of Mechanics and Maritime Sciences

Report Number: 2022:01
Project report in the course Automotive Engineering Project
Course Code: TME-180
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

PILOT NATURALISTIC RIDING STUDY (NRS) WITH VOI E-SCOOTERS TO IMPROVE TRAFFIC SAFETY
Project report in Automotive Engineering Project
Daniel Schmidt, Fredrik Lindelöw, Gowtham Gunashekara, Rahul Rajendra Pai, Chalmers University of Technology

Abstract

Increase in traffic and emissions resulting from the rise in vehicle population and the first/last mile issue associated with the use of public transport has led to a rise in the micro-mobility market. The e-scooters take a major share of this market and can be attributed to the e-scooter rental service. Introduction of e-scooters into the traffic environment has resulted in new traffic conflicts hence, possible hazards that can lead to new types of near-crashes or crashes. To better understand how e-scooters interact with other road users and identify underlying risk factors that may lead to a crash, naturalistic data collection is a suitable tool for proactive traffic safety work. In fact, the naturalistic data offer the unique opportunity to understand the cause of crashes and the genuine road user behaviour in critical situations. In the past, naturalistic data has been collected mainly from motorized vehicles. In this pilot project an e-scooter has been equipped with a data logger that is connected to cameras and numerous on-board sensors. The sensors provide kinematic information of the e-scooter while the cameras provide a visual representation of the ride environment. Each of the hardware components is placed in a casing developed with additive manufacturing technologies. The data collected is stored locally and then can be transferred to a computer for data analysis. The data collection has been carried out in two stages with the initial one being to identify any vulnerability of the system. The second set of data collection include participant-based riding data. Graphical User Interface (GUI) has been developed to enable easier analysis and visualisation of the data. Several other tools required for data processing along with the GUI have been developed using MATLAB. These tools will enable frame by frame analysis of the ride and aid in the understanding of the cause of every critical event recorded. This thereby enables detection of the causation mechanism behind the safety-critical scenarios. The project has proven that it is fully possible to equip an e-scooter with instrumentation for naturalistic data collection. A GUI has been proven to be a necessary asset when it comes to evaluating the logged data. The results from this pilot study may be the basis for a large naturalistic data collection, that may cast light on safety of e-scooters.

Key words: Naturalistic Data, E-scooters, Vehicle Safety, Data Logger, E-scooter Safety.

Acknowledgements

We would like to thank our supervisors Giulio Bianchi Piccinini and Marco Dozza for their support and guidance during every step of the project. Special thanks for the feedback from the initial test riding of the prototype e-scooter.

Our heartfelt gratitude to our supervisor from Voi, André von Dankert for his valuable insights in the technical details of the e-scooter, his support in the hardware and software development. Your support helped us get through the difficult phase of the project with less stress. From Autoliv we would like to extend our profound gratitude towards Christian-Nils Åkerberg Boda for his valuable feedback at every stage of the project.

Special thanks to Ron Schindler for taking time and participating in the data collection during the snowing weeks of December. Thank you, Alexander Rasch, for sharing his knowledge about naturalistic data collection and introducing us to previous studies in this area.

We are grateful to Voi Technology AB as a whole for supporting the project with an e-scooter and providing a creative work environment. Thank you, Mikael Lastin, from Voi for guiding us in the initial phase of the project and welcoming us to Voi. We take this opportunity to also thank all the people in the workshop at Voi for support during the manufacturing of the prototype.

Thank you, XP-workshop, at Chalmers Johanneberg for providing 3d-printing and the tools necessary for manufacturing the prototype.

Daniel Schmidt, Gothenburg, January 2022

Fredrik Lindelöw, Gothenburg, January 2022

Gowtham Gunashekara, Gothenburg, January 2022

Rahul Rajendra Pai, Gothenburg, January 2022

Table of contents

1. Introduction 1
2. Data collection instrumentation 3
2.1 Hardware 3
2.1.1 Housing 4
2.1.2 Cameras 6
2.1.3 Raspberry Pi on board computer 7
2.1.4 J-Link 8
2.1.5 Power supply 8
2.2 Software 8
2.2.1 E-scooter control functionality 9
2.2.2 Camera recording functionalities and features 9
2.2.3 Data logging 11
3. Data collection 12
4. Data Analysis 13
4.1 Data Processing 13
4.2 Graphical User Interface 14
4.3 H264 to MP4 video conversion 16
5. Discussion, conclusion, and future work 16
5.1 E-scooter prototype 16
5.1.1 Data logger 16
5.1.2 Casing 17
5.2 Data analysis 17
5.3 Limitations and scope for future work 18
Appendix 1 – Prototype manufacturing 20
Appendix 2 – Software 22
References 28

List of Figures

Figure 1. Voi e-scooter equipped with prototype data logger 3

Figure 2. Outline of the data flow in the prototype. 3

Figure 3. a) The assembly of setup currently in use. b) The complete assembly with the case. 4

Figure 4. a) The front 3/4 perspective of the base plate. b) The rear 3/4 perspective of the base plate. 5

Figure 5. a) The J-Link mounted on the baseplate with a case. b) The Raspberry Pi assembly mounted on the J-Link case. 5

Figure 6. Housing for the power bank on the backside of the e-scooters stem. 6

Figure 7. a) Front facing camera 220° FoV lens. b) Rider facing camera with a 192° FoV lens. 6

Figure 8. View of the intersection with a 170° FoV lens and 5MP OV5647 sensor. 7

Figure 9. View of the intersection with a 220° FoV lens and 5MP OV5647 sensor. 7

Figure 10. Unlock and lock button that also stops and starts the video and data recording. 7

Figure 11. Trigger button situated on the left handlebar underneath the indicator. 7

Figure 12. Button connection to the Raspberry Pi. 8

Figure 13. Overview of the software functionalities. 9

Figure 14. Flowchart of the e-scooter control thread. 9

Figure 15. Flowchart of the camera recording thread. 10

Figure 16. Flowchart of the data logging thread. 11

Figure 17. Acceleration signal pre-filtered vs filtered. 13

Figure 18. Orientation of the co-ordinate system for the accelerometer 14

Figure 19. First iteration of the GUI with both front facing camera and rider facing camera playing simultaneously. 15

Figure 20. Second iteration of the GUI complete with the option to select front facing camera or rider facing camera and a pointer on the graph indicating the position of the video. 15

Figure 21. Third iteration of the GUI complete with forward and rewind options for the video. 15

Figure 22. GUI used to analyse data and video recordings from the participants rides. 16

Figure 23. Near crash recorded during the data collection. 18

List of Tables

Table 1. Selected variables to be studied in the project. 2

Table 2. Camera calibration parameters. 10

1. Introduction

The global sale of vehicles is growing at a rapid pace with an average annual sales of 68 million vehicles [1]. The significant rise in vehicle population has resulted in increased traffic congestion with drivers spending increasing amount of time in traffic [2], [3]. This coupled with an increasing environmental concern over the emissions from internal combustion engine-based vehicles, has led to the ubiquity of micro-mobility vehicles such as e-bikes and e-scooters. This rapid growth can be attributed to the rise in the 'scooter sharing' [4] companies. First introduced in 2017 [5] these e-scooter sharing companies now operate across various cities and provide users to ride and pay by the minute. A study conducted by L. Gebhardt et.al. [6] investigated the potential of e-scooters as a substitute for replacing car trips. The results vary depending on the geographical area and if the rider is an inhabitant in the city or visitor hence, 10% of the car trips could be replaced by e-scooters in Germany [6]. This corresponds well to the need for diversified transportation in the cities. According to a report by Trafikkontoret the cycling needs to increase by 200% until 2025 [7] a part of which can be met by micro-mobility vehicles.

With numerous cities promoting the use of public transportation, these e-scooter sharing services aim to provide a solution to the first/last mile problem associated with the use of public transport [8]. The e-motor equipped e-scooter with a battery persisting several kilometres is an attractive substitute to other modes of transport which in city limits run at an average of 15 to 20 km/h [9]. A report by NATCO indicates that in 2019 alone, a total of 86 million trips across the United States were taken on e-scooters [10] which is 123% increase as compared to the 38.5 million trips in 2018 [11]. Likewise, Sweden has seen an increase in e-scooter rental services with a total of 1.2 million users in 2020 alone [12]. The e-scooters are currently being ridden in the bicycle lanes which in Sweden is shared with pedestrians. Although the e-scooters and the rider behaviours are different as compared to the conventional bike [13], [14], the current road legislations treat both to be similar. The bicycle lanes not being intended for the speeds at which the e-scooters travel in along with the increased number of users has led to new types of traffic conflicts with traditional bicyclists [14]. With the rise in the number of riders, the topic of safety of e-scooter riders and the other road users is now even more important. An increase in the number of e-scooter rider injuries and hospitalisations has been observed [15] with a very high number of head injuries [16], [17]. A study conducted in Sweden indicates 83% of the injured e-scooter riders had been involved in single crashes [18]. According to the Voi annual safety report, 34% of severe injury crashes occur during night-time, where environmental factors such as limited visibility may be a cause [19]. A study by Trivedi et. al. states that 18% of the e-scooter related hospitalisations are connected to alcohol intoxication [20]. However, little research has been done to understand the crash causation mechanisms or the rider behaviour that results in the crash.

Naturalistic data is recognised as a good measure to understand and address traffic safety issues [21]. The study conducted on 100-Car data indicates that for a naturalistic study, near-crashes can be used as a surrogate measure for crashes [22]. Naturalistic data as of date are mainly collected from cars and trucks with some studies carried out on bicycles [23], [24]. While some studies have been conducted to understand the naturalistic behaviour of e-scooterist, they are collected externally using video cameras mounted on cars and infrastructures [14], [25]. To be able to work proactively on the development of safety technologies and adapt the regulations, one must understand why the interactions happen, hence gathering naturalistic data with sensors on the e-scooter is a requirement. This will enable a better understanding of the scenarios that

lead to safety-critical situations between the e-scooters and other road users. Inspired by naturalistic bicycle studies carried out hitherto, this project intends to develop a prototype e-scooter that would enable naturalistic data collection. A Graphical User Interface (GUI) developed as a part of the project will enable an easier analysis of the data collected. The project as a whole will enable a better understanding of how e-scooter can be equipped for naturalistic data collection and also assess the quality of the data collected. The prototype and the data analysis tools will serve as a platform based on which a large-scale naturalistic data collection can be carried out.

For conducting naturalistic data analysis understanding the variables to be analysed becomes ever so important, helping in instrumentation and thereby data collection. The variables selected to be studied as a part of the project are listed in Table 1. These help in the selection of relevant data collection equipment to be installed for instance a rider facing camera can help identify the number of riders, rider behaviour while the kinematic part of the rider behaviour can be identified using the onboard sensors from the Internet of Things (IoT) of the e-scooter. The variable type 'categorical' indicate the variables that can be divided into subgroups. The 'time stamp' category indicates the variable can be analysed at a 1Hz rate while that of 'time series' must be analysed frame by frame.

Table 1. Selected variables to be studied in the project.

	Variable	**Variable type**	**Examples for subcategories**	
Rider behaviour	Number of riders	Categorical	1, 2, 3, 4	
	Rider behaviour	Categorical	Traffic Violation Erratic riding	Impairment
	Time of behaviour	Time series		
	Conflict partner type	Categorical	E-scooter Bicycle Pedestrian Light Vehicle	Heavy vehicle Animals Object None
	Conflict time	Time stamp		
Environment	Weather	Categorical	Sunny Rainy	Snowing Foggy
	Road surface	Time series	Asphalt Pavement	Icy Wet
	Lighting condition	Categorical	Bright Night	Dawn Dusk
	Road type	Categorical	Bicycle lane Pedestrian Lane Car lane	Car lane Parking Lanes Intersection
	Speed Zones	Categorical	Slow zone Normal operation	

2. Data collection instrumentation

Figure 1. Voi e-scooter equipped with prototype data logger

In this study, one Voi e-scooter as shown in Figure 1 has been instrumented with a front-facing camera and rider-facing camera, a J-Link, a power bank, and a Raspberry Pi 3 to enable logging and software implementations to control key features of the e-scooter. These features involved locking and unlocking the e-scooter with a button that simultaneously would start and stop all data recording. A second button has been added to allow the rider to flag any event that was perceived as safety-critical or of importance. The e-scooter has been equipped from the factory with a headlight, tail light, bell, turn indicators and various reflectors of which only the front reflector was re-positioned, to adhere to Swedish traffic laws [19]. The basic outline of the flow of data from various instrumentation is as shown in Figure 2. These components will be described in more detail in the following sections.

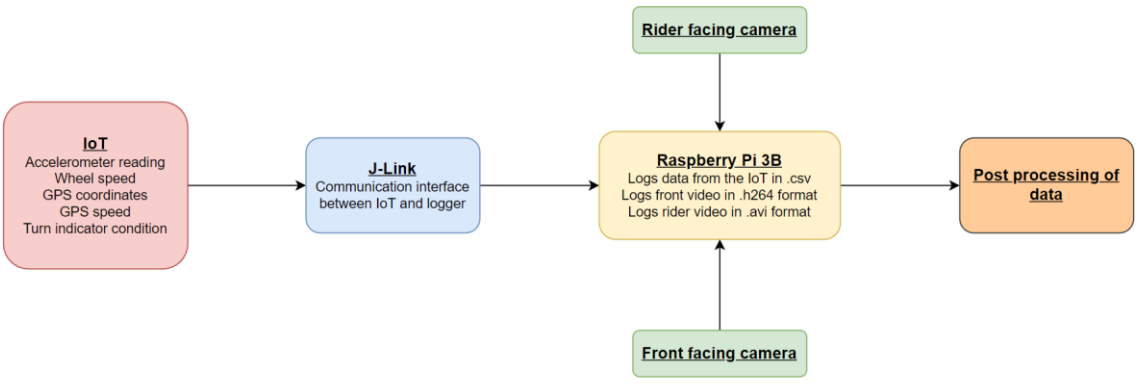


Figure 2. Outline of the data flow in the prototype.

2.1 Hardware

The hardware components consist of 14 parts, excluding wiring as listed below. The subsections below explain each component in detail. Every piece of hardware, except for the buttons,

wiring, cameras, J-Link and Raspberry Pi has been designed and manufactured in-house. The IoT unit which, is the black box located on the front side of the stem, as shown in Figure 1. Data logger setup has been mounted above the IoT unit.

1. Case for the J-link
2. J-Link
3. Case for the Raspberry Pi 3B
4. Raspberry Pi 3B
5. Front facing camera mount
6. Front facing camera
7. Baseplate
8. Main housing
9. Poly-carbonate lid
10. Power bank housing
11. Trigger button
12. Unlock/lock button
13. Rider facing camera mount
14. Rider facing camera

2.1.1 Housing

One of the design targets of the project has been to create a discrete design that would look natural on the e-scooter and at the same time be scalable to future projects hence, as simple as possible to manufacture. The design of the housing has been developed using 3d-CAD CATIA [26] and PTC Creo [27]. Four different iterations have been developed and tested. Each of the iterations can be studied in Appendix 1 Figure A1. In the final design, the hardware components are stacked on top of each other with individual casings inside the larger housing as shown in Figure 3. Outlets for connecting the rider facing camera (bottom) and the power bank (right side) have been incorporated. As presented in Figure 4, the housing has been attached to the stem of the e-scooter using a rigid baseplate. This baseplate is screwed on using already existing holes on the e-scooter. The parts are mounted together with screws for easier disassembly and the possibility to change single parts for new iterations as shown in Figure 5.

The manufacturing of the housing was performed in-house at the Chalmers Johanneberg XP workshop. The casings have been manufactured using additive manufacturing using Prusa i3 MK3s with PETG plastic filament. PETG being less brittle than PLA and easier to print than ABS is ideal for the current application.

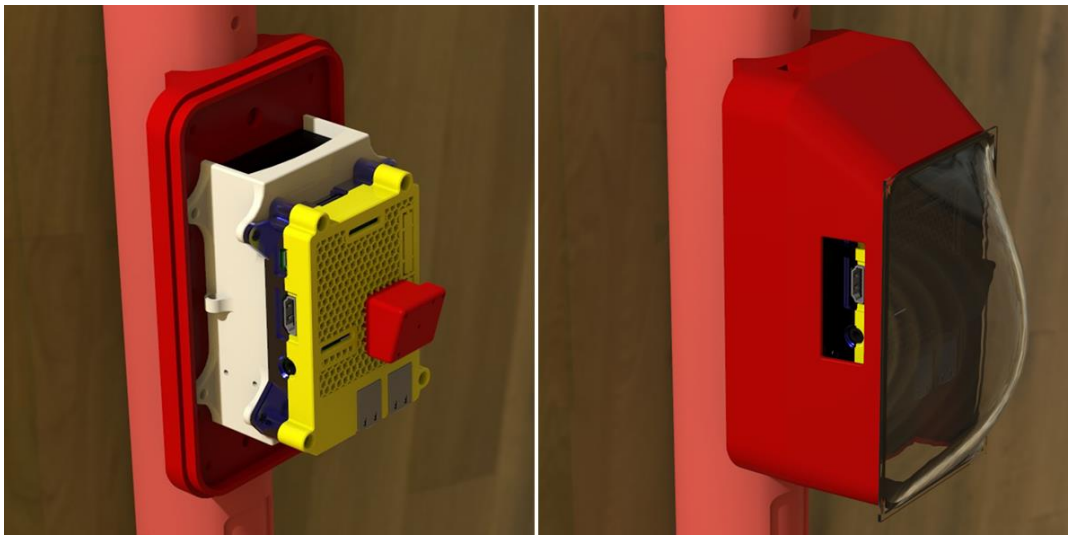


Figure 3. a) The assembly of setup currently in use. b) The complete assembly with the case.

Figure 4. a) The front 3/4 perspective of the base plate. b) The rear 3/4 perspective of the base plate.

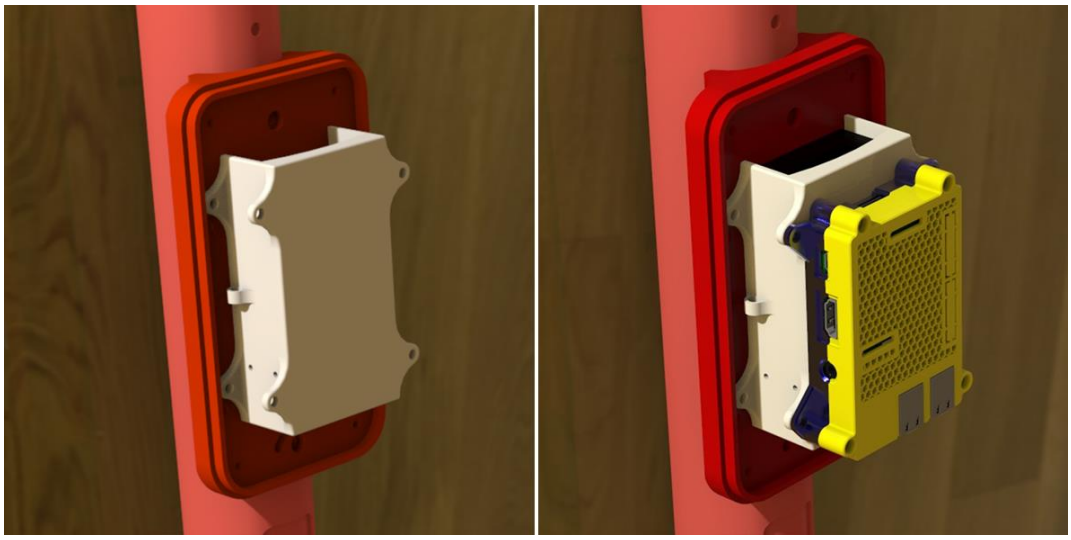


Figure 5. a) The J-Link mounted on the baseplate with a case. b) The Raspberry Pi assembly mounted on the J-Link case.

To make the entire setup splashproof while not obstructing the FoV of the camera, a case was developed such that a small gap for the camera wedge is open as shown in Appendix 1 Figure A1d. This setup failed as it is not an ideal design to manufacture using 3D printing. The final design incorporated transparent plastic which has been manufactured using polycarbonate. A mould as presented in Appendix 1 Figure A2 has been designed and manufactured using the 3D printer with ABS as the material as it can sustain higher temperatures. The polycarbonate sheet is formed to the correct shape by heating and pressing in the mould. The manufacturing process is as shown in Appendix 1 Figure A3 and A4. The heating of the polycarbonate was conducted with a heat gun hence, achieving the correct glass temperature (TG) of the polycarbonate has proven difficult and required several attempts.

A separate housing was manufactured for the power bank on the backside of the stem with access points to both the input and output slots, see Figure 6.

Figure 6. Housing for the power bank on the backside of the e-scooters stem.

2.1.2 Cameras

Numerous iterations of camera hardware have been tested to obtain a wider Field of View (FoV) and thereby get a better understanding of surroundings. The understanding of the surrounding becomes important during analysis as this provides valuable information such as the road condition, the conflict partner, the lighting condition, the direction in which the rider is looking etc. Although some of it can be obtained from interviews with the riders, the chance of them remembering it wrong highlights the importance of the camera recordings in naturalistic data collection. The road-facing camera is based on a 5MP OV5647 sensor as shown in Figure 7a. In the first iteration, the sensor had been coupled with an M9 170° FoV lens. The resulting image in one of the scenarios is as shown in Figure 8. From the video, it is evident that the vehicles on either side of the e-scooter especially when travelling in an intersection had been missing in the footage due to the smaller FoV. Therefore, the lens has been replaced with an M9 220° FoV as shown in Figure 9. The resulting image captured in the same position shown in Figure 9 presents increased visibility. With the new lens, the vehicles travelling perpendicular to the e-scooters are also captured. This can be very helpful in a sideway crash or critical incidents in intersections. The additional FoV also provides an insight into the position of the rider's hands on the brake levers.

The rider camera is added to provide information about the rider's head position and the glances during the riding. The camera can also help understand if the rider has been distracted or performing any secondary tasks while riding (e.g., using the cell phone). A 3 MP sensor coupled with a 192° FoV has been connected to the Raspberry Pi data logger using a Universal Serial Bus (USB). The sensor along with lens arrangement is as shown in Figure 7b.

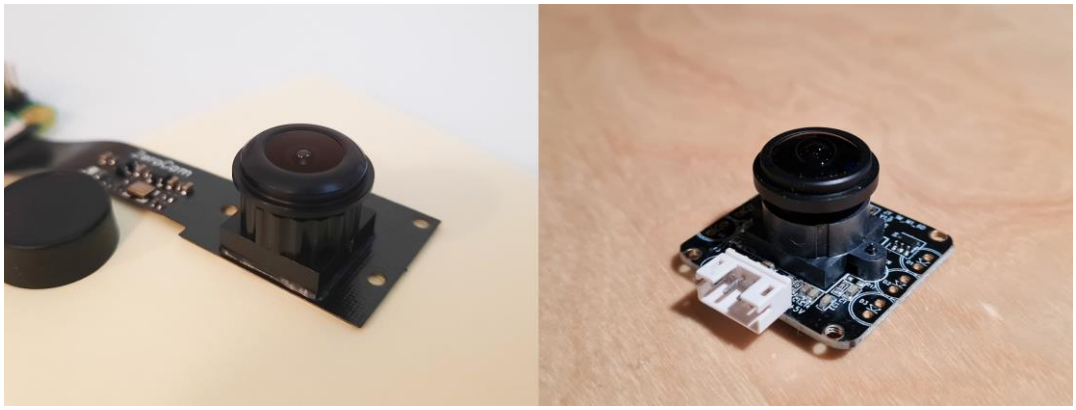


Figure 7. a) Front facing camera 220° FoV lens. b) Rider facing camera with a 192° FoV lens.

Figure 8. View of the intersection with a 170° FoV lens and 5MP OV5647 sensor.

Figure 9. View of the intersection with a 220° FoV lens and 5MP OV5647 sensor.

2.1.3 Raspberry Pi on board computer

The data logger selection has been done with the large-scale implementation in mind. This meant the logger must not only have a small form factor but also be cost-effective. Raspberry Pi zero had been tested initially. But given the J-Link software requires at least ARM Cortex-A53 the Pi zero proved to be incompatible thereby necessitating the switch to a Raspberry Pi 3B. The Pi 3 is running Raspbian Buster operating system with a desktop.

Two buttons are connected to the Pi. One of it is to lock/unlock the other is to flag the critical or any noteworthy incidents, see Figure 10 and Figure 11. The connections of the buttons with the Pi are as shown in Figure 12. The button that locks/unlocks the e-scooter as shown in Figure 10 is connected to the General-Purpose Input/Output (GPIO) pin 10. The button also controls the start and stop of logging. The trigger flag button is connected to the GPIO pin 12 and is mounted on the handlebar below the turn indicator button as shown in Figure 11.

Figure 10. Unlock and lock button that also stops and starts the video and data recording.

Figure 11. Trigger button situated on the left handlebar underneath the indicator.

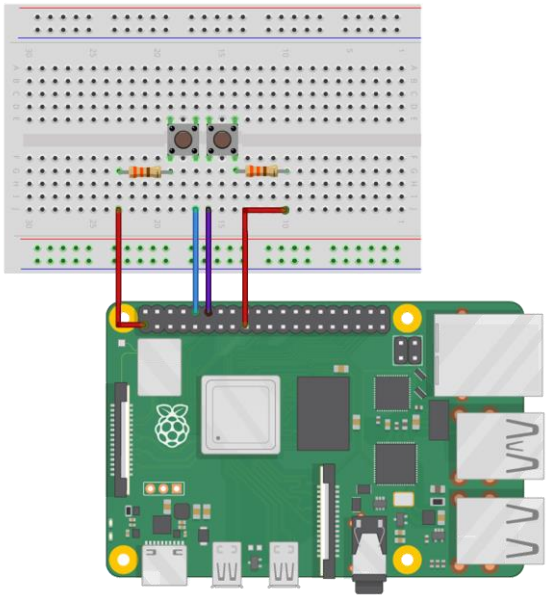


Figure 12. Button connection to the Raspberry Pi.

2.1.4 J-Link

The J-link sits between the IoT unit of the e-scooter and the Raspberry Pi, as shown in the schematic overview in Figure 2 and its physical location can be seen in Figure 5 as a black box within the white housing. The J-link carries out Real-Time Transfer (RTT) extracting data from the IoT and dumping it into the terminal in the Raspberry Pi via the USB. The 'J-link Plus' debug probe is used in the setup which is rather bulky and makes up for most of the space inside the casing. This however can be replaced with a smaller version of the probe as there is no debug operation carried out.

2.1.5 Power supply

Power has been supplied to the Raspberry Pi through a 10 000 mAh power bank which sits inside a case designed on the back of the e-scooter's stem. With the Raspberry pi having a requirement of a 5V supply [28], the average current drawn was measured to be 0.5A. This resulted in average power consumption of 2.5W see Equation 1. Efforts have been made to implement the power supply from the e-scooter's battery to the Raspberry Pi in order to exclude the power bank. However, given the e-scooter has only 3.3V and 40V lines used for the low voltage peripherals an easy to implement solution was not available. Therefore, the need for a 5V supply has been met using the power bank in the current prototype.

$$P = U * I = 5 * 0.5 = 2.5\,Watt \quad (1)$$

2.2 Software

To give functionality to the hardware, suitable for the needs and requirements of the project, several scripts have been developed in python language. The software developed can be divided into three threads [29] one being e-scooter control functionality, the second camera recording functionalities and features, and the third data logging as shown in Figure 13. The program is enabled to run on the boot of the Pi to avoid any manual intervention using Cron [30].

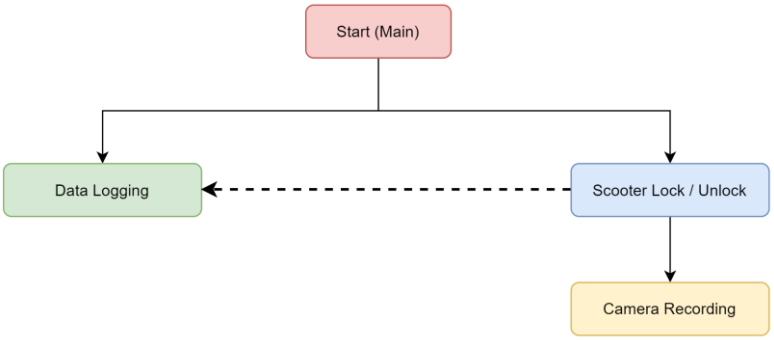


Figure 13. Overview of the software functionalities.

2.2.1 E-scooter control functionality

The e-scooter control is based on the button press and is initialized with the start of the program. The flowchart presented in Figure 14 provides the basic outline of how the button functionality has been implemented in the program. The program thread has an infinite loop that constantly tracks for the button press and comparing with the previous state of the e-scooter i.e., locked or unlocked, takes action accordingly. The unlock operation also is responsible for the start of the camera record thread while the lock operation is responsible to kill the camera record thread.

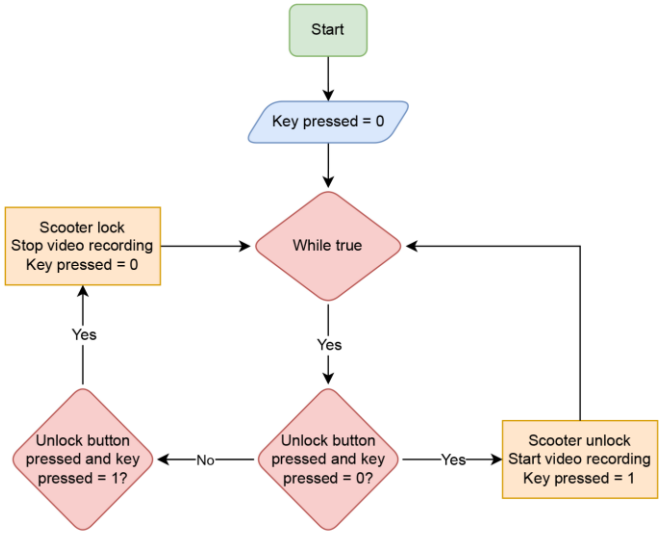


Figure 14. Flowchart of the e-scooter control thread.

2.2.2 Camera recording functionalities and features

The reason for using two cameras has been to give a visual image to complement the data to achieve a better understanding of the environment and understand the rider behaviour during a critical event. To combine video footage with numerical data it is highly important to achieve good synchronisation, meaning the timestamp in the video must correlate with the same time stamp in the data. The video files are named based on the date and time the e-scooter is unlocked with prefixes of 'Front_' and 'Rider_' added to the file names. The video is annotated by using the system clock on the Raspberry Pi and is continuously updated. When the trigger button is pressed by the rider the background of the timestamp will turn red which otherwise will be black. An exception to this is if the power supply to the Raspberry Pi is disrupted, then it defaults to the last known time since the system clock requires an internet connection to read

the correct time. Figure 15 presents the flowchart of the camera recording thread which is initialised and killed using the e-scooter functionality thread.

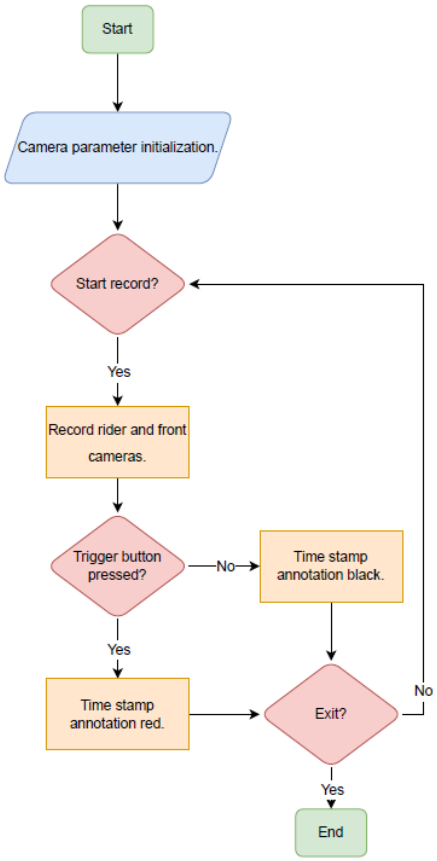


Figure 15. Flowchart of the camera recording thread.

An important distinction between the front-facing camera and the rider-facing camera is that the front-facing camera is connected via a Camera Serial Interface (CSI) cable and allows for certain features in python, while the rider-facing camera is connected through a USB port and can be accessed by FFmpeg [31] or OpenCV [32]. Given the installation of OpenCV resulted in errors, FFmpeg has been used to record the data. For the front-facing camera, the brightness and contrast have been changed to 65 and -5 respectively. This is to better account for the declining sun exposure during the winter. The rider-facing camera has been calibrated to a brightness of 100 and a contrast of -0.1, see Table 2 for a full description of parameters.

Table 2. Camera calibration parameters.

	Brightness			**Contrast**		
	Set	Default	Range	Set	Default	Range
Front facing	65	50	[0, 100]	-5	0	[-100, 100]
Rider facing	100	0	[-1000,1000]	-0.1	0	[-1,1]

The resolution and the frame rate have been a compromise between the quality of the video and the space consumed. Numerous iterations have been carried out on each of the camera modules. The resolution of the front-facing camera was configured to 720x532p with a frame rate of 30

frames per second (FPS). The rider-facing camera was recording in 640x480p with an average fps of 17. The video from the front-facing camera is recorded in the native .h264 format to avoid any extra processing required to convert it to a different format. The rider-camera being recorded using FFmpeg on the other hand is saved as a .avi file. The use of USB based camera in combination with the FFmpeg results in a lower performance as compared to the csi cable based front camera controlled using the picamera library.

2.2.3 Data logging

The data logging is carried out in a thread started and running in parallel to the e-scooter control thread. The IoT of the e-scooter sends out data as soon as the e-scooter has been unlocked. This data is received into the command line of the Pi through the J-Link debug probe. The data received starts with the string 'DATALOG:' and is detected among the lines to extract only relevant information from the command line. The timestamp based on the system clock of the Raspberry Pi is added to the data obtained from the command line. The same time stamp as mentioned in section 2.2.2 is also annotated on the video footages captured. Since both the kinematic data and camera are marked with the same clock at the same instant, the sync between them is achieved. In addition, a trigger flag is added to the data string which is based on the trigger button state i.e., 1 for pressed and 0 otherwise. Every time the e-scooter is unlocked, a new .csv file is created with the corresponding date and time as the filename; headers are added to enable easy data analysis for each of the rides. Ideally, the data is available as soon as the e-scooter is unlocked. However, there is a delay in data transmission in the practical implementation. The delay between the dataset and the video is found to be varying between 1 and 5 seconds (median 3 seconds). This delay is affected by unknown factors and is varying throughout, making it hard to compensate.

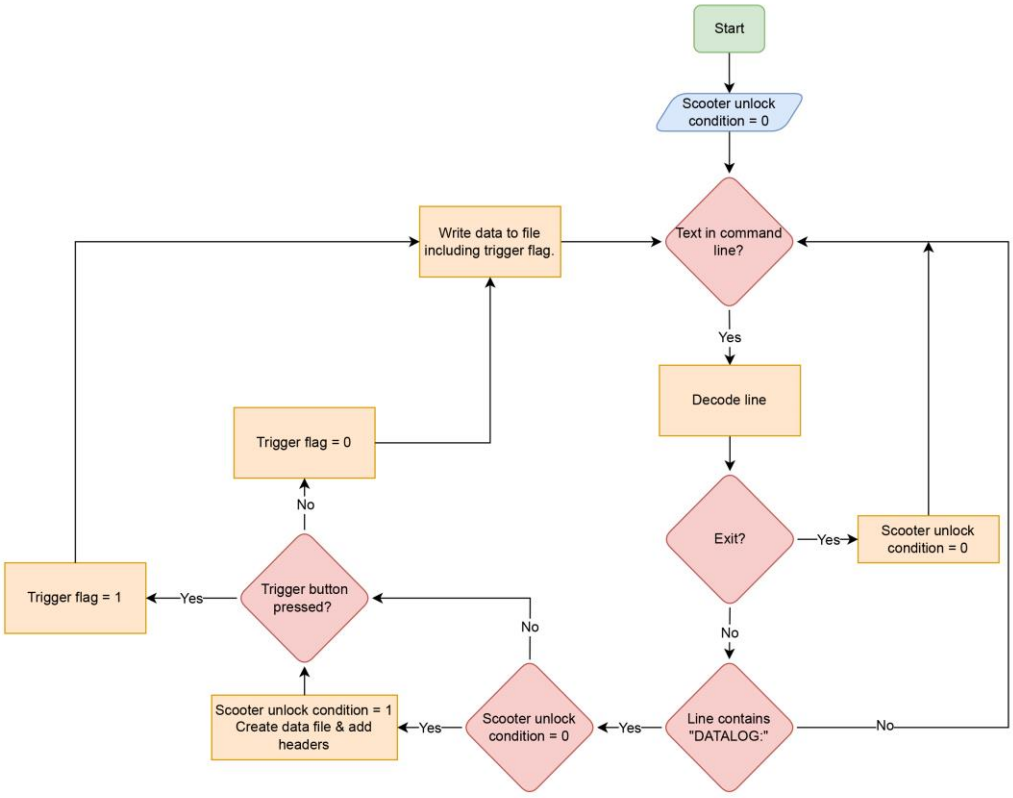


Figure 16. Flowchart of the data logging thread.

3. Data collection

During the prototype development stages, the members of the team had been collecting data to validate each of the features and instrumentation. These rides consisted of both daily commuting and long weekend trips around the city. This would cover a broader perspective of the traffic situations and expose any vulnerability of the prototype. The initial versions having no protection against water or dust had been used with extensive care so as to avoid any water damage to the devices. The final version has been tested with rides in both rain and snow. There have been no noticeable issues and the housing has proven to be splashproof. Having completed several test runs, the participant-based data collection has been conducted over the span of two weeks with three participants agreeing to use the e-scooter. Initially, five participants had been selected for the study but, due to unfavourable weather conditions, only three participants were able to take part. The participants were required to have previous e-scooter riding experience. They had been given a short introduction and then an initial short trip to get familiar with the e-scooter. The participants were asked to provide oral consent since the ride and the rider was being recorded during the journey. With all instructions and information being conveyed, the riders were offered a helmet. Two of the participants completed one trip each around Lindholmen while the third participant has been able to cover the wide range of trips spread across the city of Gothenburg.

The data is stored on a 128 GB USB stick mounted on the Raspberry Pi to ensure ease of data transfer to a computer for data analysis. The data logging is programmed to be carried out at 10Hz. But in practice, the logging rate varies between 8Hz and 12Hz. Although the GPS data is logged at the same frequency as the other data, the GPS signal update frequency is 1Hz. This means that there will be entry logs with the same GPS coordinates between the updates. It also influences the GPS speed that can have a mismatch with the wheel speed. When the e-scooter has been stored indoors or at a location where the GPS antenna doesn't have a clear signal reception, the e-scooter loses the position and fails to record both the position and the GPS speed. However, this can be solved by letting the e-scooter stand still for five minutes in an open location outside where it can recover the signal before riding it. Riding next to buildings can cause multipath issues and generate faulty coordinates. This is an issue with most GPS modules and not much can be done in this regard.

Another issue encountered after longer trips of 15 minutes was that the e-scooter does not respond to commands to lock the e-scooter. This occurs at random and there does not seem to be any physical reason for this to happen i.e., no faulty connections or corrosion of pins. To overcome this issue the rider has to manually disconnect the power bank and then connect it again and wait for the reboot to occur. Video and data would still be recorded until the power is shut off and also save it. In addition, there have been some instances where the USB stick failed to mount resulting in the data loss. This has been countered by saving it on the micro-SD card on the Pi. Although it makes for a longer process of transfer of data, it is found to be a more robust way of collecting data.

One main aspect considered as a part of the initial test runs has been to check for the synchronisation between the camera and the dataset. During the test run the scooter had been stopped and the scooter was rolled back (i.e., pushed in the reverse direction). The time stamp on the video when the scooter starts moving rearwards was noted along with the time stamp the wheel speed provided a negative value. These timestamps are found to be within a second away

from each other. Thus, indicating a good sync between the camera recording and the dataset. The GPS co-ordinate was noted for the point where the scooter had been stopped using google maps. This is compared with the GPS co-ordinate obtained from the dataset. It is observed that the points are less than 10 meters apart in cases where there has been clear reception of GPS signals. The accuracy of the magnitude of the accelerometer readings have not been carried out due to limitation in time.

4. Data Analysis

4.1 Data Processing

The kinematic data from the e-scooter is obtained at a frequency of 10 Hz. Given the data is synched with respect to the time stamp in the videos with the timestamp being in hh:mm:ss (hours:minutes:seconds) it will result in having several measurement points correlating to one second of video footage. This was corrected by averaging all the data samples over each second, thus implementing a version of a low pass filter. The acceleration in the lateral direction before and after filtering is shown in Figure 17. This approach did result in loss of detailed data, where sudden sub-second changes would be smoothened, resulting in data loss. A better approach would have been to increase the time resolution of both the video and data to include milliseconds. This would preserve important details. The issue has been resolved in the latest version of the data logging software but has not been tested due to weather issues. GPS data is logged at the same frequency as the other data, but the GPS signal update frequency is 1Hz. This means that the same GPS coordinates will be logged several times over before it updates.

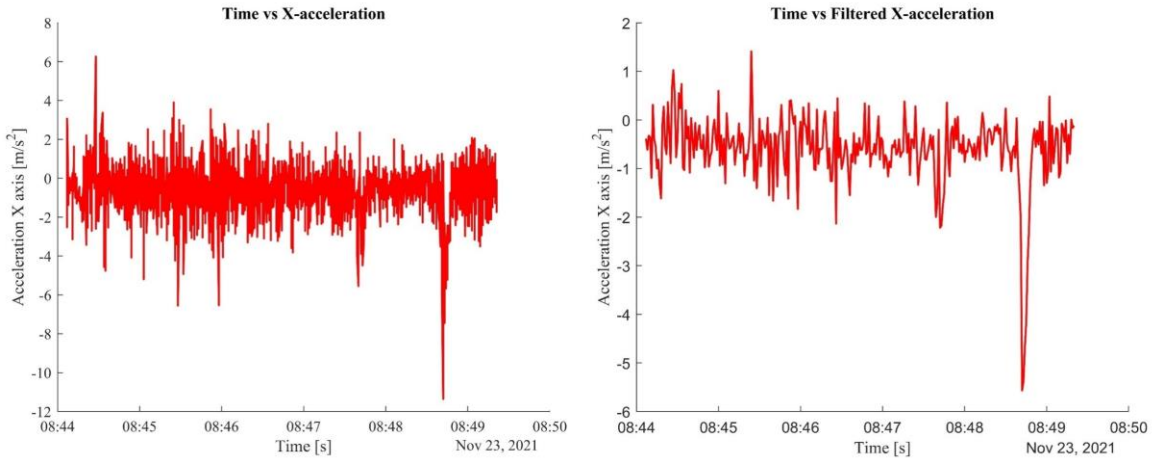


Figure 17. Acceleration signal pre-filtered vs filtered.

The inertial measurement unit is placed parallel to the e-scooter stem which is inclined at an angle of 16.3° to the horizontal. Thus, the transformation of the acceleration reading in the y-direction and the z-direction has been carried out using the Equation 2 and Equation 3. The orientation of co-ordinate system is as shown in Figure 18.

$$A_y = A_{y_{stem}} * \cos(Stem\ angle) + A_{z_{stem}} * \sin(Stem\ angle) \quad (2)$$

$$A_z = A_{z_{stem}} * \cos(Stem\ angle) - A_{y_{stem}} * \sin(Stem\ angle) \quad (3)$$

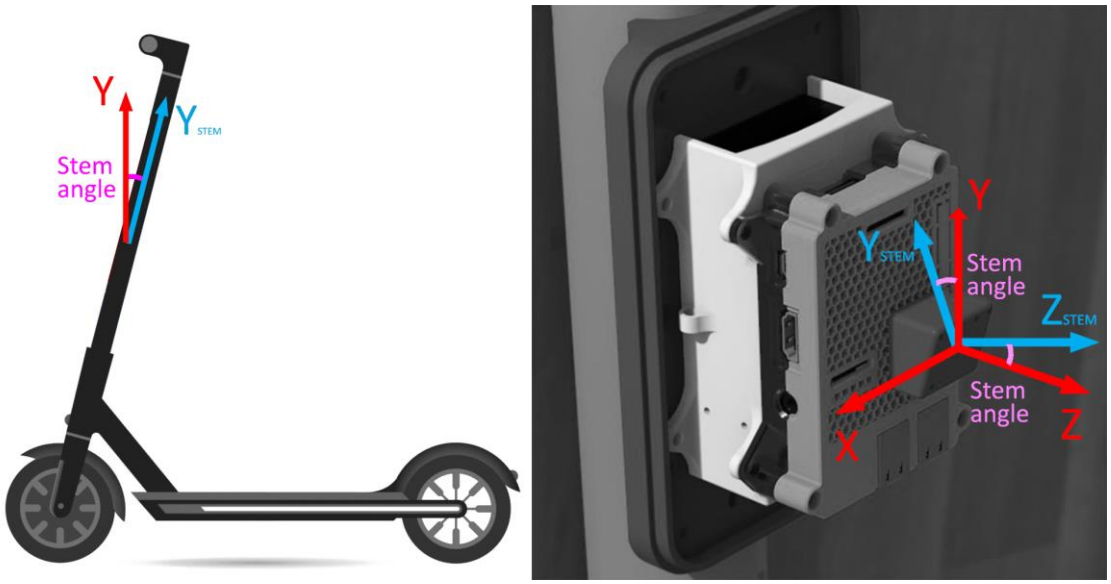


Figure 18. Orientation of the co-ordinate system for the accelerometer

4.2 Graphical User Interface

For effective analysis of the collected video and data, a GUI has been developed using MATLAB [33]. The initial version of the GUI has been developed with both the videos playing simultaneously resulting in slower playback and the requirement of higher computation power. This is presented in Figure 19. The second version of the GUI enabled the user to switch between the videos from the rider and the front-facing camera. A pointer on the graph also has been implemented which indicates the data corresponding to the video frame being played and is shown in Figure 20. The third version has an option to forward and rewind the video by 10 seconds, along with the filtering of the data as presented in Figure 21. The fourth version which has been used for carrying out data analysis has the trigger signals marked on the graph as well as the map. This will help the user to easily locate the triggers and fast forward or rewind to the specific part of the ride. The trigger signals on the graph are marked using green rectangles with a span of 10 seconds on the x-axis and the corresponding point on the map is marked using triangles. A visual representation of this GUI can be seen below in Figure 22. The user interface consists of a list of all available data sets with the associated videos, data variable to be displayed, front or rider camera selection, a play/pause button and a 10 second fast forward or backwards button.

The GUI is the tool developed specifically for the type of data collected with the e-scooter as a part of the project. With each iteration, the user-friendliness (e.g., the user having more control over the playback) has increased while keeping the application less computational intense. The tool has been used on a computer with a hexacore Intel i5-9500 CPU and 32 GB of RAM producing decent playback of the videos. Hence a computer with a similar or higher specification is recommended for the visualisation of the data. Given the front and rider facing videos have different frame rates, the program struggles to maintain sync. If the videos are recorded with the same frame rates it will be much smoother and light to run. Due to the hardware limitation of the cameras, this is not feasible in the current version of the setup.

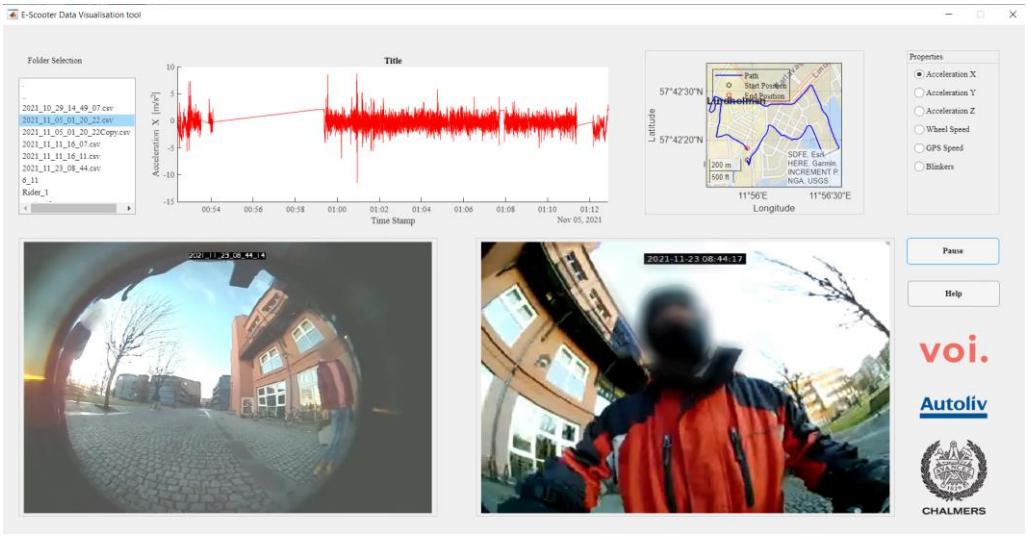


Figure 19. First iteration of the GUI with both front facing camera and rider facing camera playing simultaneously.

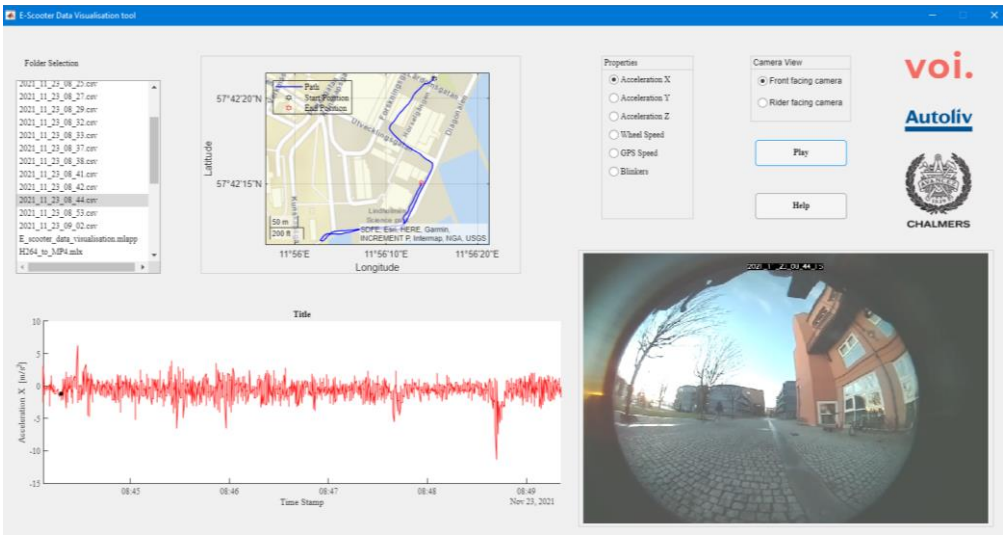


Figure 20. Second iteration of the GUI complete with the option to select front facing camera or rider facing camera and a pointer on the graph indicating the position of the video.

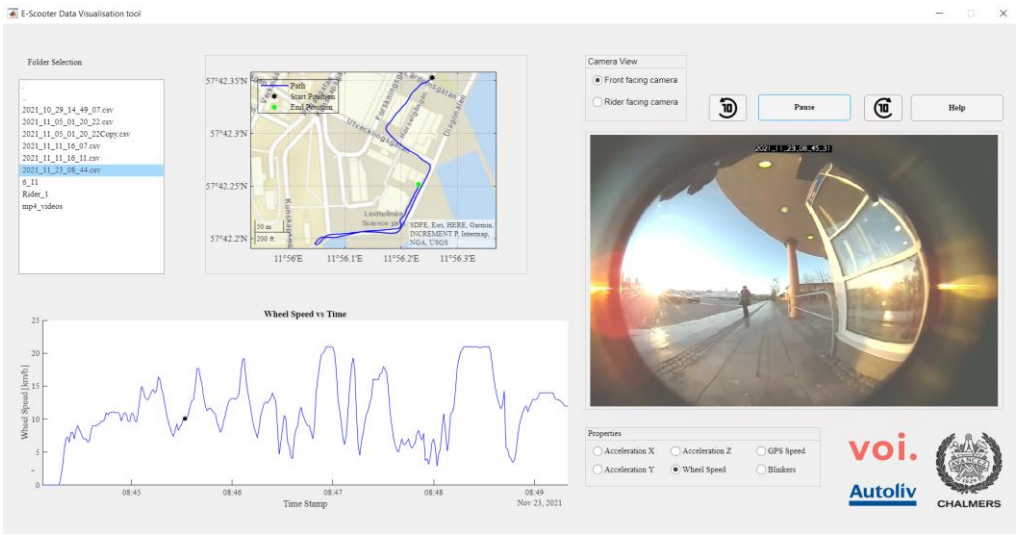


Figure 21. Third iteration of the GUI complete with forward and rewind options for the video.

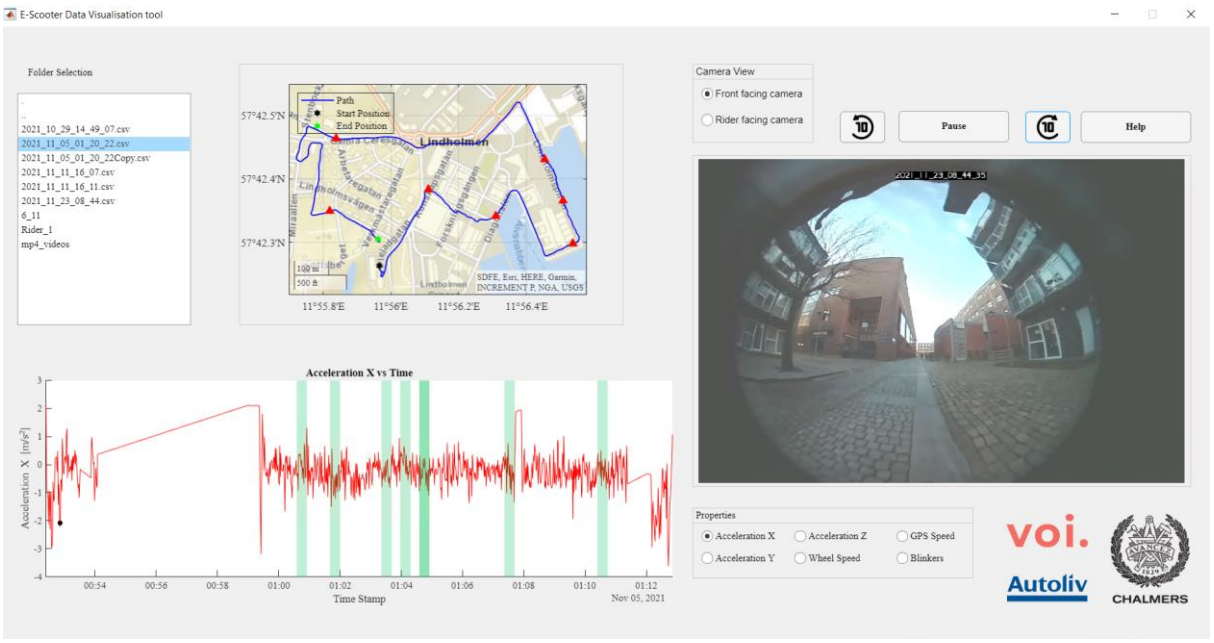


Figure 22. GUI used to analyse data and video recordings from the participants rides.

4.3 H264 to MP4 video conversion

The video recording of the front facing camera is saved in its native format '.h264'. This video format is encoded with H.264 compression and is not ideal for viewing on video players native to windows operating system. This creates the need for converting it to a widely accepted format such as .mp4 or .avi. FFmpeg is used for this conversion and is done with a MATLAB script.

5. Discussion, conclusion, and future work

The project has proven that it is fully possible to equip an e-scooter with instrumentation for naturalistic data collection. A GUI has been proven to be a necessary asset when it comes to evaluating the logged data. The data logger developed as a part of the prototype e-scooter with the various components as described in the previous section is the major outcome of the project. The GUI along with various other tools developed for the analysis of the data collected is the other outcome from the given project. The dataset collected as a part of the project is although not statistically sound but can be used for understanding the effectiveness of the prototype.

5.1 E-scooter prototype

5.1.1 Data logger

In the naturalistic bicycle study by Dozza and Werneke [34] the bicycle is equipped with a GPS, IMU, speed sensor, along with two GoPro cameras, a data logger along with an LED to sync the data. Similarly, in this study the prototype data logger receives signals from various sensors such as the wheel speed, IMU, GPS along with the two cameras connected. The advantage of using the cameras directly connected to the data logger is the ease of syncing the dataset with the camera recording which in the bicycle study was done in the data analysis stages. The button-based trigger indication used as a part of the prototype in this study is similar to the one used in the bicycle study. With the success of the BikeSAFE [35] project, the prototype e-scooter having demonstrated similar data logging capabilities one can, with high confidence, predict the effectiveness of e-scooter in naturalistic data collection. This in combination with the preliminary data logged has proven the capabilities of the prototype in effective and quality data logging. In a study by Garman et.al. [36] to understand the e-scooter dynamics a Plex VMU

900 HD Pro has been used for logging data. While the same could be implemented in this project, the cost associated with each of the loggers is high compared to the Raspberry Pi based logger, thereby drastically compromising the scalability of the prototype.

5.1.2 Casing

The previous study conducted by BikeSAFE [35] used a of the shelf box, while in this project a custom designed box was developed that could house all the hardware in a significantly more compact design.

The casings have been developed in iterations and is presented in Appendix 1. It has served its intended purpose and protected the delicate electronic equipment from rain and debris. Accessibility of the front casing is not optimal since the components were stacked on top of each other, making it difficult to reach certain IO-ports on the Raspberry Pi. Disassembling the casing requires loosening several screws and carefully peeling off the casing. The battery box has performed well.

The design and placement of the trigger button switch has proven to be effective since it has the same position as the horn button on a motorcycle which should be a natural position to press in a safety critical situation. The unlock/lock button position was chosen to be far away from the trigger-button to decrease the chances of it being pressed accidentally or confused with the trigger-button.

5.2 Data analysis

During the data collection, one near crash has been recorded. This has been identified by means of an interview with the participant. Another way to identify these events is based on the trigger flag indicated by the rider. In addition, a kinematic based trigger can also be implemented by introducing a threshold for parameters such as the acceleration in the x-direction. When the logger senses any parameter beyond the threshold set a flag is marked. The flag indicates the occurrence of an event of interest and further can be verified by discussions with the rider. The data can be used in various types of analyses such as identification of causation mechanism, driver modelling and statistical analysis. The following paragraphs explain the application of the dataset in detail for each of the three types of data analysis.

Several studies conducted based on naturalistic data have opted for an odds ratio based statistical analysis to analyse the effect of exposure to a certain aspect on the outcome [24], [37]–[39]. Bálint et. al. carried out the odds ratio-based study to understand the influence of multitasking on crash risks [37] while Owens et. al. conducted a similar investigation for understanding the association of cell phone use and risk of a crash in cars [39]. Given this e-scooter project is implemented on a larger scale and there is a bigger dataset collected, one can carry out a similar analysis. For instance, the video sample of a near-crash scenario recorded is illustrated in Figure 23. This can be annotated based on the parameters listed in Table 1 such as road surface was icy, lighting condition was bright with the rider riding alone and no conflict partners. For every such critical incident, two random baseline events are selected from the data of the same rider. Similar to the critical events, the baseline events are annotated for the variables listed in Table 1. With multiple such data being noted, an odds ratio can be calculated to understand the possibility of an icy road surface resulting in a crash or likewise other parameters listed. However, several near-crashes or crashes are required for this analysis.

Another application of the data would be to understand the crash causation mechanism. Several studies conducted formerly have proven the effectiveness of the use of naturalistic data in crash mechanism estimation [40]–[42]. The study conducted by Piccinini et. al. uses naturalistic data to investigate the rear-end crash causation mechanism of commercial vehicles in China [41]. Similarly, from an extensive data collection with the e-scooters, one can carry out a similar analysis. Case in point, from the camera recording of the ride shown in Figure 23, it is evident that the rider brakes before entering the corner and then tilts the e-scooter into the corner while throttling thereby losing the rear end of the e-scooter and resulting in a near-crash scenario. Similarly, annotating every safety-critical scenario in a large data set will help understand and conclude the mechanisms generally leading to such an event. For instance, if the majority of the events involve riders using throttle taking the corner or leaning beyond a certain angle it can be safe to conclude that these are the mechanisms that lead to a crash. Having a throttle input and the brake input into the dataset will be an added advantage and help analyse these datasets better.

Figure 23. Near crash recorded during the data collection.

Research carried out previously with naturalistic datasets such as the Strategic Highway Research Program 2 (SHRP2) data [43]–[47] have illustrated the application of naturalistic data in driver behaviour estimation and counterfactual simulation. With the dataset collected on the e-scooter similar to the one collected in the SHRP2 study, a driver model can be developed based on the head movements, reaction time, and the motion trajectory of the e-scooterist. This model can then be used for the development of active safety systems such as a warning system on the e-scooter. These can also help implement safety measures on trucks or cars based on the trajectory estimation of an e-scooterist. The information on the rider behaviour will help in better development of driver assisting systems on the cars or trucks with some limited scope of being implemented on the e-scooter itself.

5.3 Limitations and scope for future work

The electronic components used, J-link and Raspberry Pi 3 are unnecessary bulky and if the project would have had information about the smaller J-link and used a Raspberry Pi Zero two that is considerable smaller. The cables are placed outside the stem and is protected by shrink wrap. This results in several access holes through the casing making, water penetration through the case more likely. This can possibly be solved by having the cables inside the stem and the handlebar. This requires major modification of the e-scooter stem hence has not been carried out in the project.

As of now, all the peripherals and Raspberry Pi are powered by a power bank which both counters the goal of a discrete design package and the scalability ability. Power can be taken from one of the e-scooters lines but needs then to be stepped down to 5 V for the Raspberry Pi. The bulky J-link used as a part of the data logger can also be replaced with a J-link mini edu or can be eliminated with the use of SPI communication.

The python script as described earlier is running different threads. This is to separate the different functions and avoid overlapping issues. But this results in not just higher computational power but also complications in identifying the part of the code that failed. There perhaps may be a better way to do it but given the limited knowledge of the team this was not implemented.

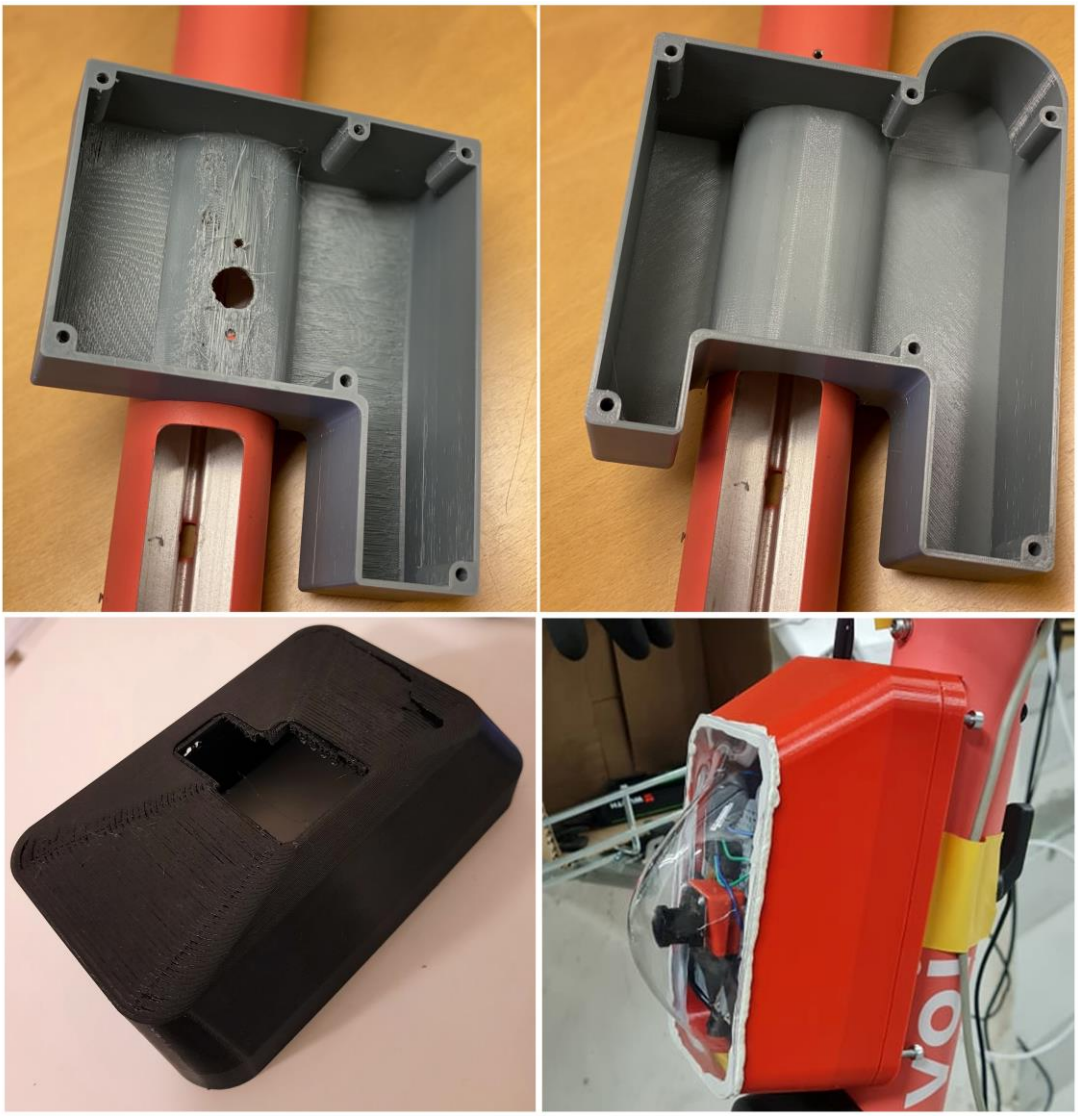
As this is a 15-credit course several time-constraints have been put on the project which limited the scope. Due to limited time and changing weather conditions, only three trusted participants have partaken in the study. This may have resulted in some bias of the dataset but, given it is a pilot study, and main goal has been to ensure the possibility of data collection, the bias has been neglected.

Another form of limitation imposed on the study were in the form of budgetary restrictions which resulted in a large quantity of in-house development of hardware, compromised the quality (i.e., frame rate and resolution) of the USB camera as well as the onboard computational power. Access to classified information regarding the communication scheme of the e-scooter would have drastically changed the design approach of both hardware and software.

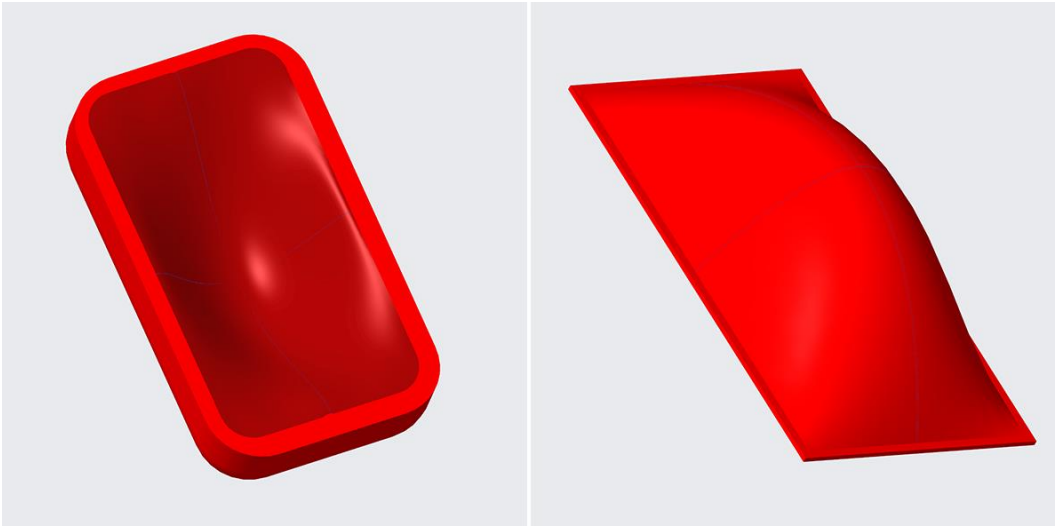
Since no extensive data collection was carried out it was not possible to fully evaluate the GUI in its ability to be used as an analysis tool. The final iteration of the design of the main housing can be considered sleek when compared with its previous versions, but still fall short of a theft proof setup and easy to manufacture solution required for a large-scale introduction. This also affects the scalability of the setup since the components used can be considered too expensive when scaled up and the local storage method will result in difficulties to retrieve any stored data. The next improvement would then be to implement cloud storage so that all data recorded will be automatically synched to a cloud-based drive and thus eliminate the need for physical data retrieval.

One final point of improvement would be to eliminate the need for a physical unlock/lock button and instead utilize the Voi app to control the fundamental functions of the e-scooter. The app is the standard way of renting and accessing a normal e-scooter in the wild.

Appendix 1 – Prototype manufacturing



A1. a) First iteration of housing. b) Housing with cable management. c) Housing with enclosure with a cut-out for the camera. d) Final housing with transparent lid.



A2. a) Mould for the housing. b) Press for moulding.

A3. Heating of the polycarbonate front facing transparent casing.

A4. Forming the front facing polycarbonate in the mould.

Appendix 2 – Software

Plot the data collected from the e-scooter

The following script can be used to plot various signals received from the e-scooter.

Please use the GUI to visualise the data along with the video.

This is an alternative way to visualise the data and customise the plots as required.

```
clear all;
clc;
close all;
addpath(genpath(fileparts(which("Data_Processing.mlx"))));

% Import data from file
data_file_name = '2021_11_23_08_44.csv';
```

All the data available from the e-scooter are extracted with relevant labels here.

Use these to plot additional graphs in the next section of the script.

```
% Extracts the data using a custom function 'extract_data' which can be
% found in the function files tab
file_data = extract_data(data_file_name);

% Time in seconds
time_of_run = datetime(file_data.Time_stamp,...
    "InputFormat","yyyy_MM_dd_HH_mm_ss");
time_elapsed = seconds(time_of_run-time_of_run(1));

% CPU time during execution (in millisecond) converted to seconds
program_time_stamp = (file_data.Run_Time-file_data.Run_Time(1))./1000;

[Acceleration_filtered_X,Acceleration_filtered_Y,Acceleration_filtered_Z,...
    filtered_time_set, filtered_wheel_speed] =
func_acceleration_filter(file_data);

% Extract accelerometer readings
Acceleration_X1 = Acceleration_filtered_X./1000000;
Acceleration_Y1 = Acceleration_filtered_Y./1000000;
Acceleration_Z1 = Acceleration_filtered_Z./1000000;

Stem_angle = 16.3; %Degrees

% Stem angle compensation
Acceleration_X = Acceleration_X1;
Acceleration_Y = Acceleration_Y1*cosd(Stem_angle)+ ...
    Acceleration_Z1*sind(Stem_angle);
Acceleration_Z = Acceleration_Z1*cosd(Stem_angle)- ...
```

```
    Acceleration_Y1*sind(Stem_angle);

% Speed measured at the wheel
Wheel_Speed = file_data.wheel_speed;

% Speed from GPS in IoT
GPS_Speed = file_data.GPS_Speed;

% GPS coordinates (Longitude)
GPS_Long = file_data.Longitude;

%GPS coordinates (Latitude)
GPS_Lat = file_data.Latitude;

% Trigger point index
Index_trigger=find(file_data.Trigger_Signal==1);

% To avoid plotting the multiple points in the same location due to button
% being pressed long
Index_trigger = [Index_trigger(1); ...
    Index_trigger(find(diff(Index_trigger)>1)+1)];
```

Generating the plots:

```
% X-ACCELERATION VS TIME
figure()
hold on
plot(filtered_time_set, Acceleration_X,'r-','LineWidth',1.25)
plot(filtered_time_set, smoothdata(Acceleration_X,'gaussian',7),...
    'k-','LineWidth',1.25)
title('Time vs X-acceleration','FontName','Times')
xlabel('Time [s]','FontName','Times')
ylabel('Acceleration X axis [m/s^2]','FontName','Times')
legend('Original','Smoothened','FontName','Times')
plot_incident(get(gca,'YLim'),Index_trigger,time_of_run)

% Y-ACCELERATION VS TIME
figure()
hold on
plot(filtered_time_set, Acceleration_Y,'r-','LineWidth',1.25)
plot(filtered_time_set, smoothdata(Acceleration_Y,'gaussian',7), ...
    'k-','LineWidth',1.25)
title('Time vs Y-acceleration','FontName','Times')
xlabel('Time [s]','FontName','Times')
ylabel('Acceleration Y axis [m/s^2]','FontName','Times')
legend('Original','Smoothened','FontName','Times')
plot_incident(get(gca,'YLim'),Index_trigger,time_of_run)


% Z-ACCELERATION VS TIME
```

```
figure()
hold on
plot(filtered_time_set, Acceleration_Z,'r-','LineWidth',1.25)
plot(filtered_time_set, smoothdata(Acceleration_Z,'gaussian',7), ...
    'k-','LineWidth',1.25)
title('Time vs Z-acceleration','FontName','Times')
xlabel('Time [s]','FontName','Times')
ylabel('Acceleration Z axis [m/s^2]','FontName','Times')
legend('Original','Smoothened','FontName','Times')
plot_incident(get(gca,'YLim'),Index_trigger,time_of_run)

% WHEEL SPEED VS TIME
figure()
hold on
plot(time_of_run ,Wheel_Speed,'r-','LineWidth',1.25)
plot(filtered_time_set, filtered_wheel_speed,'b','LineWidth',1.25)
xlabel('Time [s]','FontName','Times')
ylabel('Wheel Speed [km/h]','FontName','Times')
title('Time vs Wheel Speed','FontName','Times')
plot_incident(get(gca,'YLim'),Index_trigger,time_of_run)

% WHEEL SPEED & GPS SPEED
figure()
hold on
plot(time_of_run, Wheel_Speed,'r-','LineWidth',1.25)
plot(time_of_run-seconds(8), GPS_Speed,'b-','LineWidth',1.25)
title('Wheel and GPS Speed','FontName','Times')
xlabel('Time [s]','FontName','Times')
ylabel('Speed [km/h]','FontName','Times')
legend('Wheel Speed','GPS Speed','FontName','Times')
plot_incident(get(gca,'YLim'),Index_trigger,time_of_run)

% GPS COORDINATES ONTOP OF LOCAL MAP
figure()
geoplot(GPS_Lat(GPS_Lat>0),GPS_Long(GPS_Long>0),'b','LineWidth',1.15)
hold on
%Finds the first and last non-zero GPS entry
geoplot(GPS_Lat(find(GPS_Lat>0,1)),GPS_Long(find(GPS_Long>0,1)),'gh',...
    'MarkerFaceColor','g')
geoplot(GPS_Lat(find(GPS_Lat>0,1,'last')),...
    GPS_Long(find(GPS_Long>0,1,'last')),'mh','MarkerFaceColor','m')
legend('Path','Start Position','End Position','Location','NorthWest',...
    'FontName','Times','color','none')
geobasemap streets

% Plotting the location of trigger on the map
figure()
geoplot(GPS_Lat(Index_trigger),GPS_Long(Index_trigger),'rh', ...
    "MarkerFaceColor",'r',"MarkerSize",10)
hold on
```

```
geoplot(GPS_Lat(Index_trigger),GPS_Long(Index_trigger),'rh', ...
    "MarkerFaceColor",'r',"MarkerSize",10)
geobasemap streets

% Function to plot the trigger flags
function []=plot_incident(lim_values,Index_trigger,time_of_run)
    if ~isempty(Index_trigger)
        for i=1:length(Index_trigger)
            start_incident = time_of_run(Index_trigger(i))-seconds(10);
            end_incident = time_of_run(Index_trigger(i))+seconds(10);
            fill([start_incident,start_incident,end_incident,...
               end_incident],[lim_values,flip(lim_values)],[0 0.751 0.36],...
                'FaceAlpha',0.25,'EdgeColor',"none",'HandleVisibility','off')
        end
    end
end

% Function to extract the data from the .csv file into a usable format
function data_set = extract_data(filename)
    opts = delimitedTextImportOptions("NumVariables", 12);
    opts.DataLines = [2, Inf];
    opts.Delimiter = ["", " "];

    % Specify column names and types
    opts.VariableNames =["Time_stamp", "Trigger_Signal", "Run_Time", "ax",...
        "ay", "az", "wheel_speed", "GPS_Speed", "Indicator", "Longitude",...
        "Latitude", "Var12"];
    opts.SelectedVariableNames =["Time_stamp", "Trigger_Signal",...
        "Run_Time", "ax", "ay", "az", "wheel_speed", "GPS_Speed",...
        "Indicator", "Longitude", "Latitude"];
    opts.VariableTypes =["string", "double", "double", "double", "double",...
       "double", "double", "double", "double", "double", "double", "string"];

    % Specify file level properties
    opts.ExtraColumnsRule = "ignore";
    opts.EmptyLineRule = "skip";
    opts.ConsecutiveDelimitersRule = "join";
    opts.LeadingDelimitersRule = "ignore";

    % Specify variable properties
    opts = setvaropts(opts, "Var12", "WhitespaceRule", "preserve");
    opts = setvaropts(opts, "Var12", "EmptyFieldRule", "auto");

    % Import the data
    data_set = readtable(filename, opts);

    % Clear temporary variables
    clear opts
end
```

Convert the video collected from the e-scooter

This program is to convert a .h264 format video collected on the e-scooter to .mp4 format to view the videos

Pre-requisites: FFmpeg installed

Instructions:

Step 1: Paste all the h264 files into the folder 'Video' alongside this script.

Step 2: Run this script to convert the files into MP4

Installing the FFmpeg on windows can be done by following the instructions in the link below

https://www.wikihow.com/Install-FFmpeg-on-Windows

```
clc
clear all
close all
addpath(genpath(fileparts(which("H264_to_MP4.mlx"))));

% PLEASE SPECIFY THE ID OF THE RIDER HERE
folder_name_gen = "11_11";
```

Obtaining the list of files in the folder and if it is h264 we convert it to mp4

Note: The mp4 files will be saved in the folder within mp4 folder and the h264 will be moved to a h264 folder. A copy of the mp4 videos will be done into IoT_Data folder as well to enable the GUI functionality.

```
file_list = dir;

if exist('mp4_video','dir')~=7
    mkdir("mp4_video")
end

if exist('h264_files','dir')~=7
    mkdir("h264_files")
end

mkdir('./h264_files/',folder_name_gen)

ffmpegDir = 'C:\FFmpeg'; % Specify the FFmpeg installation location here if
its different

command_part = ['"' fullfile(ffmpegDir, 'bin', 'ffmpeg.exe') '" -r 30 -i '];
```

```
for i=1:length(file_list)
    file_name = file_list(i).name;
    file_names_split = split(file_name,'.h264');
    if length(file_names_split)==2
%         cmd = ['"' fullfile(ffmpegDir, 'bin', 'ffmpeg.exe') '" -r 30 -i
%         vid.h264 -vcodec copy myvid.mp4 &']; ,' 1 > NUL 2 > NUL'
        cmd=[append(command_part,char(file_names_split(1)),...
            '.h264 -vcodec copy
"%cd%/mp4_video/"',char(file_names_split(1)),...
            '.mp4 &',' 1 > NUL 2 > NUL')];
        [status, message] = system(cmd);
        pause(2)
        movefile(file_name,append('./h264_files/',folder_name_gen));
    end
end

% Once the conversion is succesful, the batch of files are then put in a
% folder within the mp4_video folder
if exist ("status", 'var')==1
    disp('successfully executed')
    cd mp4_video
    mkdir(folder_name_gen);
    mp4_file_list = dir;
    for i=1:length(mp4_file_list)
        mp4_file_name = mp4_file_list(i).name;
        mp4_file_names_split = split(mp4_file_name,'.mp4');
        if length(mp4_file_names_split)==2
           copyfile(mp4_file_name,...
               append("../../IoT_Data/Dataset/mp4_video/",folder_name_gen));
        end
    end
end

% If a compression of video is needed for any reason uncomment the
% following line and run the script.

% video_compression_algorithm
```

References

[1] R. Young, “Global Auto Sales Slid Again in September as a Production Turn-Around Remains Elusive | Post,” *Scotiabank*, Oct. 29, 2021. https://www.scotiabank.com/ca/en/about/economics/economics-publications/post.other-publications.autos.global-auto-report.october-28--2021.html (accessed Nov. 20, 2021).

[2] B. Moya-Gómez and J. C. García-Palomares, “The impacts of congestion on automobile accessibility. What happens in large European cities?,” *J. Transp. Geogr.*, vol. 62, pp. 148–159, Jun. 2017, doi: 10.1016/J.JTRANGEO.2017.05.014.

[3] Y. S. Chang, Y. J. Lee, and S. S. B. Choi, “Is there more traffic congestion in larger cities? - Scaling analysis of the 101 largest U.S. urban centers-,” *Transp. Policy*, vol. 59, pp. 54–63, Oct. 2017, doi: 10.1016/J.TRANPOL.2017.07.002.

[4] S. Shaheen, A. Cohen, and I. Zohdy, “Shared Mobility: Current Practices and Guiding Principles,” *Fhwa-Hop-16-022 2.*, no. Washington D.C., p. 120, 2016.

[5] A. J. Hawkins, “The electric scooter craze is officially one year old — what’s next? - The Verge,” *The Verge*, Sep. 20, 2018. https://www.theverge.com/2018/9/20/17878676/electric-scooter-bird-lime-uber-lyft (accessed Nov. 20, 2021).

[6] L. Gebhardt, C. Wolf, and R. Seiffert, “‘I’ll Take the E-Scooter Instead of My Car’—The Potential of E-Scooters as a Substitute for Car Trips in Germany,” *Sustain.*, vol. 13, no. 13, 2021, doi: 10.3390/su13137361.

[7] Trafikkontoret, “Trafik- och resande- utveckling 2020,” Gothenburg, 2020. [Online]. Available: https://goteborg.se/wps/wcm/connect/17c52d1f-3a52-42a0-8d39-a6b7360dafed/TRU_2020_slutversion.pdf?MOD=AJPERES.

[8] H. Wang and A. Odoni, “Approximating the Performance of a ‘Last Mile’ Transportation System,” *https://doi.org/10.1287/trsc.2014.0553*, vol. 50, no. 2, pp. 659–675, Sep. 2014, doi: 10.1287/TRSC.2014.0553.

[9] S. Gössling, “Integrating e-scooters in urban transportation: Problems, policies, and the prospect of system change,” *Transp. Res. Part D Transp. Environ.*, vol. 79, p. 102230, Feb. 2020, doi: 10.1016/J.TRD.2020.102230.

[10] “Shared Micromobility in the U.S.: 2019 | National Association of City Transportation Officials.” https://nacto.org/shared-micromobility-2019/ (accessed Nov. 07, 2021).

[11] “Shared Micromobility in the U.S.: 2018 | National Association of City Transportation Officials.” https://nacto.org/shared-micromobility-2018/ (accessed Nov. 07, 2021).

[12] “E-Scooter-sharing - Sweden | Statista Market Forecast.” https://www.statista.com/outlook/mmo/mobility-services/e-scooter-sharing/sweden#users (accessed Nov. 07, 2021).

[13] D. García-Vallejo, W. Schiehlen, and A. García-Agúndez, “Dynamics, Control and Stability of Motion of Electric Scooters,” *Lect. Notes Mech. Eng.*, pp. 1199–1209, 2020, doi: 10.1007/978-3-030-38077-9_139.

[14] J. Todd, D. Krauss, J. Zimmermann, and A. Dunning, “Behavior of Electric Scooter Operators in Naturalistic Environments,” *SAE Tech. Pap.*, Apr. 2019, doi: 10.4271/2019-01-1007.

[15] N. K. Namiri, H. Lui, T. Tangney, I. E. Allen, A. J. Cohen, and B. N. Breyer, “Electric Scooter Injuries and Hospital Admissions in the United States, 2014-2018,” *JAMA Surg.*, vol. 155, no. 4, pp. 357–359, Apr. 2020, doi: 10.1001/JAMASURG.2019.5423.

[16] T. K. Trivedi *et al.*, “Injuries Associated With Standing Electric Scooter Use,” *JAMA Netw.*

Open, vol. 2, no. 1, pp. e187381–e187381, Jan. 2019, doi: 10.1001/JAMANETWORKOPEN.2018.7381.

[17] M. Aizpuru, K. X. Farley, J. C. Rojas, R. S. Crawford, T. J. Moore, and E. R. Wagner, "Motorized scooter injuries in the era of scooter-shares: A review of the national electronic surveillance system," *Am. J. Emerg. Med.*, vol. 37, no. 6, pp. 1133–1138, Jun. 2019, doi: 10.1016/J.AJEM.2019.03.049.

[18] H. Stigson, I. Malakuti, and M. Klingegård, "Electric scooters accidents: Analyses of two Swedish accident data sets," *Accid. Anal. Prev.*, vol. 163, p. 106466, Dec. 2021, doi: 10.1016/J.AAP.2021.106466.

[19] Voi, "Safer streets with shared micromobility," 2021. [Online]. Available: https://www.voiscooters.com/wp-content/uploads/2021/06/Voi_Safety-Report_2021.pdf.

[20] B. Trivedi, M. J. Kesterke, R. Bhattacharjee, W. Weber, K. Mynar, and L. V. Reddy, "Craniofacial Injuries Seen With the Introduction of Bicycle-Share Electric Scooters in an Urban Setting," *J. Oral Maxillofac. Surg.*, vol. 77, no. 11, pp. 2292–2297, Nov. 2019, doi: 10.1016/J.JOMS.2019.07.014.

[21] M. Dozza and N. P. Gonzalez, "Recognizing Safetycritical Events from Naturalistic Driving Data," *Procedia - Soc. Behav. Sci.*, vol. 48, pp. 505–515, Jan. 2012, doi: 10.1016/J.SBSPRO.2012.06.1029.

[22] T. A. Dingus, M. T. McGill, S. G. Klauer, and F. Guo, "Evaluating the Relationship Between Near-Crashes and Crashes: Can Near-Crashes Serve as a Surrogate Safety Metric for Crashes?," 2010. Accessed: Nov. 07, 2021. [Online]. Available: https://www.nhtsa.gov/document/evaluating-relationship-between-near-crashes-and-crashes-can-near-crashes-serve-surrogate.

[23] M. Dozza and A. Fernandez, "Understanding bicycle dynamics and cyclist behavior from naturalistic field data (November 2012)," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 376–384, Feb. 2014, doi: 10.1109/TITS.2013.2279687.

[24] M. Dozza and J. Werneke, "Introducing naturalistic cycling data: What factors influence bicyclists' safety in the real world?," *Transp. Res. Part F Traffic Psychol. Behav.*, vol. 24, pp. 83–91, May 2014, doi: 10.1016/J.TRF.2014.04.001.

[25] F. W. Siebert, M. Ringhand, F. Englert, M. Hoffknecht, T. Edwards, and M. Rötting, "Braking bad – Ergonomic design and implications for the safe use of shared E-scooters," *Saf. Sci.*, vol. 140, p. 105294, Aug. 2021, doi: 10.1016/J.SSCI.2021.105294.

[26] CATIA, *V5-6R2019*. Paris, France: Dassault Systèmes, 2019.

[27] PTC Creo, *Creo Parametric 5.0.0.0*. Boston, USA: PTC.

[28] "Raspberry Pi 3 Model B+." Accessed: Dec. 20, 2021. [Online]. Available: https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf.

[29] "Threading — Python 3.10.1 documentation." https://docs.python.org/3/library/threading.html (accessed Dec. 20, 2021).

[30] "Scheduling Tasks on reboot with Cron." https://www.raspberrypi.com/documentation/computers/using_linux.html#running-a-task-on-reboot (accessed Dec. 24, 2021).

[31] "FFmpeg," *FFmpeg team*. https://ffmpeg.org/ (accessed Dec. 01, 2021).

[32] "OpenCV." https://opencv.org/ (accessed Dec. 01, 2021).

[33] MATLAB, *9.11.0.1809720 (R2021b)*. Natick, Massachusetts: The MathWorks Inc., 2020.

[34] M. Dozza, J. Werneke, and A. Fernandez, “Piloting the Naturalistic Methodology on Bicycles,” Sep. 2012, Accessed: Dec. 24, 2021. [Online]. Available: https://publications.lib.chalmers.se/records/fulltext/local_166062.pdf.

[35] M. Dozza, “BikeSAFE | SAFER – Vehicle and Traffic Safety Centre at Chalmers.” https://www.saferresearch.com/projects/bikesafe (accessed Dec. 24, 2021).

[36] C. Garman, S. G. Como, I. C. Campbell, J. Wishart, K. O’Brien, and S. McLean, “Micro-Mobility Vehicle Dynamics and Rider Kinematics during Electric Scooter Riding,” *SAE Tech. Pap.*, vol. 2020-April, no. April, Apr. 2020, doi: 10.4271/2020-01-0935.

[37] A. Bálint, C. A. C. Flannagan, A. Leslie, S. Klauer, F. Guo, and M. Dozza, “Multitasking additional-to-driving: Prevalence, structure, and associated risk in SHRP2 naturalistic driving data,” *Accid. Anal. Prev.*, vol. 137, p. 105455, Mar. 2020, doi: 10.1016/J.AAP.2020.105455.

[38] S. G. Klauer, F. Guo, B. G. Simons-Morton, M. C. Ouimet, S. E. Lee, and T. A. Dingus, “Distracted Driving and Risk of Road Crashes among Novice and Experienced Drivers,” *N. Engl. J. Med.*, vol. 370, no. 1, pp. 54–59, Jan. 2014, doi: 10.1056/NEJMsa1204142.

[39] J. M. Owens, T. A. Dingus, F. Guo, Y. Fang, M. Perez, and J. Mcclafferty, “Title Crash Risk of Cell Phone Use While Driving: A Case-Crossover Analysis of Naturalistic Driving Data,” Blacksburg, Virginia, Jan. 2018. Accessed: Dec. 16, 2021. [Online]. Available: https://aaafoundation.org/wp-content/uploads/2018/01/CellPhoneCrashRisk_FINAL.pdf.

[40] H. Hekla Eiríksdóttir, “Quantitative analysis of rear-end crash causation mechanisms based on naturalistic crash data,” Göteborg, Sweden, Apr. 2016. Accessed: Dec. 16, 2021. [Online]. Available: https://publications.lib.chalmers.se/records/fulltext/235332/235332.pdf.

[41] G. Bianchi Piccinini, J. Engström, J. Bärgman, and X. Wang, “Factors contributing to commercial vehicle rear-end conflicts in China: A study using on-board event data recorders,” *J. Safety Res.*, vol. 62, pp. 143–153, Sep. 2017, doi: 10.1016/J.JSR.2017.06.004.

[42] L. Pipkorn and G. Bianchi Piccinini, “The role of off-path glances: A quantitative analysis of rear-end conflicts involving Chinese professional truck drivers as the striking partners,” *J. Safety Res.*, vol. 72, pp. 259–266, Feb. 2020, doi: 10.1016/J.JSR.2019.12.023.

[43] A. Ghasemzadeh and M. M. Ahmed, “Quantifying regional heterogeneity effect on drivers’ speeding behavior using SHRP2 naturalistic driving data: A multilevel modeling approach,” *Transp. Res. Part C Emerg. Technol.*, vol. 106, pp. 29–40, Sep. 2019, doi: 10.1016/J.TRC.2019.06.017.

[44] A. Das, A. Ghasemzadeh, and M. M. Ahmed, “Analyzing the effect of fog weather conditions on driver lane-keeping performance using the SHRP2 naturalistic driving study data,” *J. Safety Res.*, vol. 68, pp. 71–80, Feb. 2019, doi: 10.1016/J.JSR.2018.12.015.

[45] M. M. Ahmed and A. Ghasemzadeh, “The impacts of heavy rain on speed and headway Behaviors: An investigation using the SHRP2 naturalistic driving study data,” *Transp. Res. Part C Emerg. Technol.*, vol. 91, pp. 371–384, Jun. 2018, doi: 10.1016/J.TRC.2018.04.012.

[46] J. Bärgman, C. N. Boda, and M. Dozza, “Counterfactual simulations applied to SHRP2 crashes: The effect of driver behavior models on safety benefit estimations of intelligent safety systems,” *Accid. Anal. Prev.*, vol. 102, pp. 165–180, May 2017, doi: 10.1016/J.AAP.2017.03.003.

[47] J. Bärgman, V. Lisovskaja, T. Victor, C. Flannagan, and M. Dozza, “How does glance behavior influence crash and injury risk? A ‘what-if’ counterfactual simulation using crashes and near-crashes from SHRP2,” *Transp. Res. Part F Traffic Psychol. Behav.*, vol. 35, pp. 152–169, Nov. 2015, doi: 10.1016/J.TRF.2015.10.011.