CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Behavior Based Secondary Task Action Detection In Driver Monitoring Systems

Master's thesis in Computer science and engineering

Martin Bergström

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Behavior Based Secondary Task Action Detection In Driver Monitoring Systems

Martin Bergström

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Martin Bergström

Internal supervisor: Sjoerd Hendriks, Department of Computer Science and Engineering
External supervisor: Robert Lower, Applied IT University of Gothenburg
Company supervisor: Victor Brandt, Smarteye
Examiner: Staffan Björk, Department of Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Behavioral Detection of Actions Over Time - Temporal Behavior Based Action Detection

Martin Bergström
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Driver distraction is one of the leading causes of road accidents and fatalities in traffic, both for novice and experienced drivers. Due to this, legislation has started to pinpoint the development and usage of systems to detect and prevent this kind of behavior known as secondary tasks in the context of driving. Some secondary tasks are particularly dangerous, such as texting. For the car system to effectively be able to assist the driver in reducing such behavior, driver monitoring systems are being researched and developed. While there are many different approaches to monitoring a humans behavior, the most common one is to use cameras that feeds the video stream into machine learning models trained to recognize and identify different behaviors. The scope of this thesis covers the steps of defining phone usage in the context of driving, collecting data in a simulator, preprocessing the data and training machine learning models to be able to predict the behavior of the driver. The research questions concerns the challenges in predicting human behavior, which signals are most important in doing so and how it is possible to model the dimension of time. The framework for the implementation of the project is a hybrid approach using the double diamond structure in combination with Human-Centred AI principles and a classical machine learning workflow.

Keywords: Computer, science, computer science, engineering, project, thesis.

Acknowledgements

This endeavour would not have been possible without Sjoerd Hendriks at the Department of Computer Science and Engineering at Chalmers University of Technology, who has been the internal supervisor of this project. Your guidance and support have been crucial, I can not thank you enough.

I also would like to thank Victor Brandt, who has been the company supervisor of this project. Your guidance has been indispensable, your warm humour made me feel very welcome at Smarteye and you have also taught me to truly love pandas.

And Robert Lowe, at the University of Gothenburg Department of Applied Information Technology and RISE. Without your unyielding support over the years, I would never have found myself in this position. Words can not describe my gratitude.

And lastly, to all the wonderful people at Smarteye who have lent me their knowledge and helped me along the way. I will be forever grateful.

Martin Bergström, Gothenburg, 2024-06-11

Contents

1

Introduction

Driver distraction is one of the leading causes of road accidents and unsafe driving, both in novice and experienced drivers [1]. It is estimated that 36% of crashes caused by driver distraction would have been avoided if the driver was not distracted [2]. Due to this, legislation has started to pinpoint the development and usage of systems to detect and prevent this kind of behavior, with the European Union creating regulatory requirements [3], and EuroNCAP [4]) adding points to its safety rating, based on the implementation and performance of such systems. Driving distractions can be separated into short and long distractions, defined as glances away from the road. Long glances, i.e. >2 seconds are estimated to double the risk of a car crash. This can be separated further into specific classes of distractions, of which phone use is a major class [5]. Depending on what the driver is doing on their phone, it may be either a long or a short glance. It is deemed necessary to find alternative ways than just facial monitoring to be able to monitor the usage of a phone during driving [6]. In order to reduce the probability that phone usage is not recognized by the system, the phone needs to be tracked even when it is out of range of the system's sensors. For example, when the camera view of the phone is occluded by other objects but is still in use.

These requirements are still in the early stages, but will after 7 July 2026 be a requirement for every new car sold in the EU. Similar regulations are in effect in North America, China and Japan [7]. While the specific requirements are, as mentioned above, still vague, it is reasonable to assume that the pursuit of the Vision Zero [8] will produce the need for the cars system to have a thorough and comprehensive understanding of the driver's state and attention. Driver monitoring is an emerging field, in the sense that there is no well-defined and absolute solution to the problem.

An advanced driver distraction warning compliant system (ADDW) is a system that assists the driver in paying attention to the task of driving and warns the driver when he or she is distracted. One of the most commonly used and least invasive technologies to meet the ADDW requirement [9] is the solution that uses a behavioral approach. This usually means that one or more cameras are placed to observe the driver's yawning, blink speed, blink frequency, percentage of eye closure (PERCLOS) for a certain amount of time, gaze region, head pose and orientation, which with the help of machine learning models is able to infer the state of the driver. Other solutions include the physiological approach, which is using physiological sensors attached to the driver's body, and the vehicle-based approach, which is utilizing the

data streams of the vehicle system's computer such as acceleration, steering and braking [10][11]. It is not uncommon that several approaches are combined in order to gain a richer understanding of the driver's state [12].

A system that is able to discern between the many secondary tasks that the driver might perform would allow it to delicately warn the driver about the current distraction. Different distractions and secondary tasks call for different forms of warning feedback from the system. For example, if the driver is engaging in phone interaction, effective feedback will potentially decrease the risk of an accident, while ineffective feedback might further increase the level of distraction and cognitive workload [13]. To give the system the means necessary to assist the driver in paying attention to the task of driving and to be able to guide the driver's attention back to the task of driving when the situation calls for it, or to adapt to a diminished level of attention by for example increasing the distance to cars ahead while using adaptive cruise control, a good understanding of the driver's current state of attention is necessary. This means that the data available to the system must be comprehensive and precise, which can only be achieved through a carefully designed monitoring system and via how the drivers state is defined in the end points of such systems. State-of-the-art systems of today perform well on this task, but there are scenarios where they do fall short. Action classes like phoning, for example, vary a lot, and there are many different scenarios where systems fail and produce false negatives. While the definition of a task like phone interaction might seem obvious to a human being, the complexity of the sensory signals that will allow the system to recognize the fine granularities and diversity of phone interaction must be explored and thoroughly understood by the designer in order to one day deliver a system that unanimously understands the driver. One possible solution to this is by fusing different models.

1.1 Research problem

The research problem of this thesis concerns driver monitoring systems, and how they can be improved to better identify secondary driving tasks and distractions. Currently, the driver monitoring field of research lacks a clear definition of phone usage. There is in particular a difference in opinion regarding when different subtasks start and end. For example, when the driver is reading and responding to a text message on their phone while driving, there is ambiguity whether or not this entire event should be considered a continuous longer distraction or rather a series of shorter distractions as the driver often returns their focus towards the road several times during the event. While this might seem trivial, it becomes obvious that clear definitions are essential if they are to be used to inform a warning system of the driver's distraction state over time.

RQ1: *What characteristics should be considered when defining phone usage in the context of driving from the perspective of driver distraction warning system?*

RQ1-a: *Which sensory signals can be important in identifying and classifying phone usage in the context of driving?*

RQ1-b: *Which phone related secondary tasks are most prone to produce false nega-*

tives?

RQ2: *How may time be represented in filtering sensory signals?*

RQ3: *Does the dimension of time improve the performance of models predicting secondary tasks?*

1.2 Stakeholders

This planning report is part of a thesis report of the M.Sc. Interaction Design and technologies program at Chalmers University of Technology and encompasses 30 credits. The project is done during spring of 2024. The supervisor of this project is Sjoerd Hendriks from Chalmers University of Technology. The external supervisor is Robert Lowe from University of Gothenburg/RISE and the company supervisor is Victor Brandt from Smarteye. Staffan Björk is the examiner for the project and will grade the thesis report.

The collaborator of this project is Smarteye, which is a worldwide company from Gothenburg working with research and development within automotive and behavioural research. In later years, the focus has been on driver monitoring technology that understands, supports and predicts human behavior. One current goal of Smarteye is to provide a product that meets the upcoming safety recommendations [4] and laws [9].

1.3 Expected results

The expected results of this thesis project is two-fold. Firstly, to provide insight into how different human actions and behavior patterns can be defined and understood as data, and secondly, how this data can be collected and modelled. Furthermore, a detailed list of the definition of phone usage in the context of driving will be produced, including potential edge-cases that are challenging or not possible to model. The definitions will then be further explored from the perspective of how they may be experienced through the sensors of the driver monitoring system.

1.4 Limitations

The insights that the machine learning models in driver monitoring systems provide is only half the interaction loop. In the end product, the system will use those insights in order to assist the driver in focusing on the task of driving by providing feedback and recommendations during states of distraction and drowsiness. This is, however, outside of the scope of this report and the project will focus solely on the sensing part of the system, and how such a system is implemented. Furthermore, the data collection will use convenience sampling for the data collection, which will result in models that are not able to generalize to the real world.

1.5 Ethical considerations

A major part of Human-AI interaction are the many ethical implications that come with integrating AI algorithms into the systems that we use in our everyday life. Below follows three aspects that may relate to a project of this scope and focus.

The data A common surface level ethical aspect of building models out of data generated by humans is the fact that it is socially intrusive and potentially a breach of privacy. There is legislation in place to protect the individual, which can make data collection challenging. The only data that is being collected and saved will be tabular data in csv files that does not contain any image data or anything else that might be connected to an individual. It is therefore anonymous by default.

Life saving technology If an AI infused system used in a high risk environment like, for example, driving proves to better protect the lives of the people who can afford it, it becomes a class issue. If the technology of the future are not able to protect and serve humanity as a whole, it will only increase the divide in society. The issue may become a diffusion of responsibility, where none of the developers and researchers feel any responsibility over what is created and for whom.

Diverse data To an extent, the variation of humans that are involved in the data collection to feed and build a model will often prove the limit on the types of humans that model will be able to assist and be useful for. While being enormously important, this goes beyond the color of skin. Different people from different cultures from different places do things differently. For example, if the system is trained to identify and warn about the usage of a phone while driving, and successfully does so and thus saves lives, it will only be able to do so for the type of phone usage it will be able to observe in the data set that it is built from. It is therefore necessary to have the conversation, and to not make any claims that a model is performing well universally. While this project may have to choose its participants from a convenient sample group, it shall not assume that the model produced will be able to generalize well in the real world.

2

Background

The requirements mentioned above are vague, but what is clear is that the system must be able to track the driver's attention in order to provide warnings when attention is waning. This can in some ways be considered a wicked problem, by its property of not yet having a definitive formulation 3.1. Looking ahead in an attempt to predict future requirements, it is not far-fetched to imagine that a requirement to track states over time will emerge and be a milestone along the path to a complete monitoring system. This could help the system in not producing false negatives when objects of focus, for example a phone, leave the sensory reach of the system and continue to occupy the driver's attention. One common way to track attention is by the use of eye tracking and head pose estimation, which in this report will be referred to as a driver monitoring system (DMS). Another common system used in this context is a cabin monitoring system (CMS), which incorporates a camera that is placed further from the driver in order to get a broader view of the driver and the rest of the cabin. Instead of tracking the drivers eyes and head pose, it instead uses the body pose of the drivers upper body, among other measurements. While the papers mentioned below do not always explicitly use the terms DMS and CMS, the different systems will be categorized as such based on their placement and specific use cases. Observing the driver using the video stream of a vision-based system is a behavioral approach, and provides the system with insights into where the driver's attention is focused and states like drowsiness and fatigue. On the highest level, this solves the problem of informing the system of the user's engagement with the task of driving based on information that can be derived by observing the driver's face, body and objects in view.

2.1 State-of-the-art

Below follows an overview of different driver and cabin monitoring systems. The research mentioned is far from exhaustive, but provides a general overview of the work that has been done in the field and how the systems are set up on a technical level.

Figure 2.1: Camera view from a typical DMS camera [14].

Figure 2.2: DMS camera position in simulator

2.2 DMS

In essence, a DMS consists of a camera that is placed in a position that allows it to observe the driver's face from all angles associated with regular driving, which is commonly behind the steering wheel [15] [14] above the center panel [16] [17] or attached to the windshield [18]. The type of camera depends on the software side requirements, and the most common type for DMS is either a near-infrared (NIR) camera coupled with an IR light source that is used to improve the quality of the video frames captured by the NIR camera [14] [16] or a regular visual-light RGB camera [15] [18]. Both camera types have strengths and weaknesses, mainly concerning light conditions. While a NIR camera is able to perform well in low light conditions due to the system producing the light that is necessary, it is sensitive to direct sunlight [19], and for a regular visual-light camera, it is the other way around.

The use cases of a DMS varies, and while every solution strives to track the state of the driver there are many ways to do that. One common approach is to first locate the driver's face in the recorded frame using a face detection algorithm like the Viola-Jones face detection algorithm [15] [18] or a facial landmark detector [16] in order to segment and track the drivers face in real time. The face image data is then fed into a machine learning model that detects features and outputs either binary or multi-class classifications. While multi-class outputs are most common [15] [14] [16] [17] with classifications like gaze locations [14] [17] those locations like for example navigation system and windshield center [14] can be separated into distracted and not distracted states based on whether or not they imply that the driver is focusing on the driving or is conducting a secondary task [14]. In other words, many systems yield multi-class classifications that can be parsed as binary classifications. Furthermore, the DMS can also be used to locate objects in view like a phone [20] or to recognize the driver's facial expression [16]. While those parameters do not give a definitive classification on whether the driver is distracted or not, they can be useful proxies for the driver's level of distraction from the task of driving.

2.3 CMS

Similar to the DMS, the CMS has many use cases, but usually involves estimating the drivers body pose and identifying objects that are being used by the driver. This could be a phone, food or a cigarette. A CMS camera is placed further away from the driver than the DMS camera, in order to be able to observe the entire upper body of the driver and the surrounding space, sometimes even the entire cabin. Common placements are adjacent to the rear-view mirror [22], on the dashboard [23] [24] or attached to the middle of the windshield [25]. Similar to the DMS, the type of camera used depends on what kind of analysis is being done on the software side. A Kinect camera is often used [22] [25], which provides signals such as RGB color image and depth image. A Kinect camera delivers depth images by a technique called time of flight, which works similarly to an ultrasonic distance sensor but with IR light instead of ultrasonic sound [26]. Using IR in the CMS comes with the same

Figure 2.3: Camera view from a typical CMS camera [21].

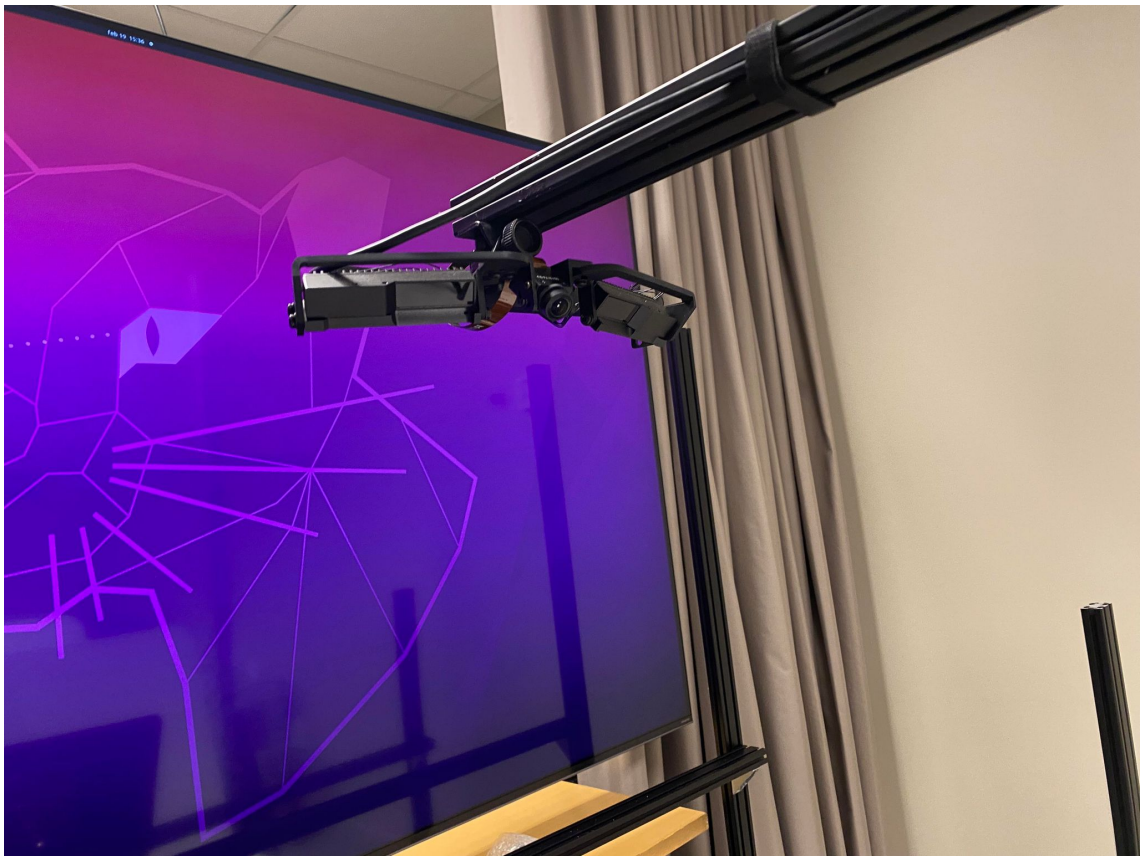

Figure 2.4: CMS camera position in simulator

pros and cons as in the DMS, with the ability to perform well in low light conditions and with sensitivity to direct sunlight. Other systems incorporate a regular RGB camera [23] [24], which thanks to advances in deep neural network performance delivers a more cost effective solution.

The most common machine learning architecture type that is being used in the CMS classification pipelines are variations of a CNN model [27]. The implementations vary. It could for example be the use of well known architectures like the VGG16 model [23], combining a CNN model and a random forest algorithm [24] or by adding pre-processing steps using different image segmentation techniques before feeding it to the CNN [25]. Another approach, which adds the dimension of time, is a recurrent neural network (RNN). This allows the system to base a classification on several frames of video data, which yields a higher performance on recognizing certain tasks [22]. A common part of CMS solutions is to create a skeletal body pose estimation, which can be used to classify actions defined by the angular relationship between joints [22] [24].

Similar to the DMS, outputs from different CMS algorithms vary from binary to multi-class classifications. The binary classification relates to attention status like drowsiness and whether or not the driver is distracted from the task of driving [25]. The multi-class classifications provide outputs like drivers activity i.e. checking rear view mirrors, interacting with the infotainment system or texting on a phone [22] [23] [24]. It is worth noting that several of the output classes, i.e. a drivers' actions, can only be recognized over several frames of the video stream [22]. Since movement is not captured by a still image, it is not possible for a frame-by-frame model to correctly predict the difference between actions like moving a phone and holding a phone. It is therefore not uncommon that models include a temporal aspect of monitoring, often realized through either an RNN [22] as mentioned above or an LSTM layer [28]. The LSTM approach is more modular than the use of an RNN, and is less dependent on a vast dataset for training.

2.4 Fusion

The state-of-the-art DMS/CMS models of today perform well in providing the system with a good understanding of the drivers state, actions and current focus. But there are edge cases where the systems fall short. While not explicitly acknowledged in the papers mentioned above, it is noticeable that there lie challenges in recognizing certain tasks that feature occluded objects and in particular when occluded objects recurrently occupy the driver's attention by long and short glances. Using phone interaction as an example, the driver might have the phone in their lap, outside of the view of both the DMS and CMS camera, and repeatedly shift their gaze from the road to their lap. This might not be recognized as phone interaction by the system, and will subsequently fly under the radar as a false negative or be interpreted as minor attention deviations short enough to not invoke an intervention from the ADDW system.

One proposed solution to increase the perceptive depth of the DMS and CMS is to

$$Y^{prod} = \prod_{i=1}^{3} \hat{Y}_i,$$
$$Y^{sum} = \sum_{i=1}^{3} \hat{Y}_i,$$
$$Y^{mean} = \arg\ \text{mean}(\hat{Y}_1, \hat{Y}_2, \hat{Y}_3),$$
$$Y^{max} = \arg\max(\hat{Y}_1, \hat{Y}_2, \hat{Y}_3).$$

Figure 2.5: Formulas for product fusion, sum fusion, average fusion and max fusion.

fuse the two types of systems. There are several methods to do this, and can be separated into out-of-the-box methods (OOTB) and learning based methods. As the name implies, the latter requires training.

2.5 Out-of-the-box fusion

OOTB methods can be utilized in multi-stream architectures where several models output multi-modal classifications of features in for example a fully connected soft-max layer. The feature maps of the different classifiers are combined using different computations and generate a generalized feature map that can be used for a final classification [29].

Product fusion is done by multiplying the feature maps element-wise [30].

Sum fusion returns the sum of the combined feature maps. Average fusion computes the mean value of the N feature maps [31].

Average fusion computes the mean value of the N feature maps [31].

Max fusion or max-pooling uses the max values in each of the N feature maps [32].

Since the four approaches above require no training, are easy to implement and work out-of-the-box, they are all very suitable candidates for evaluation assuming that the outputs of the DMS and CMS model are compatible. If this is not the case, it is still possible to build feature maps using output signals generated by both systems.

2.6 Learning based fusion

Learning based fusion methods are more complex than the OOTB methods, and can be defined by sensors used, extracted features, the underlying models that are fused, the fusion component and the outputs.

One learning based fusion implementation is the work by Ohn-bar et al [33]. Two cameras, one placed on the panel in front of the driver and one behind the driver in the middle of the cabin. From the DMS placed camera pose and landmarks cues are

extracted using a linear kernel support vector machine (SVM), and from the cabin camera hand cues. The combined cues are fed into a hierarchical SVM and outputs a classification of the driver's activity i.e. hand on wheel, gear or instrument cluster [33].

A similar solution is done by Wu et al [34], where a multi-feature fusion network based on pose estimation is proposed. In this setup, a camera is placed in a CMS position observing the driver and the driver's pose, hand features and global features are extracted using CNNs for the hand and global features and a multi-layered perceptron (MLP) for the pose feature.

Another approach, including the temporal dimension, is proposed by Kose et al [35]. Similar to the work done by Wu et al [34] the video stream of CMS placed camera is used to provide the inputs to the architecture. Optical flow frames are computed using pairs of consecutive frames that are used to represent spatial movement. An RGB frame and several optical flow frames are then concatenated, which is in other words feature level fusion, and then used for the classification in a final softmax layer. The outputs of the networks are action classifications such as safe driving, talk to passenger and drinking.

3

Theory

In the following theory section some important subjects and concepts will be described, and how they relate to this thesis work.

3.1 A wicked problem

The concept of "wicked" problems emerged in the late 1960's, during a time when old social structures and constructs were being questioned. The common way to solve problems was by attempting to look at the available facts rationally, and to then make an informed decision. This attitude stemmed from the idea of efficiency and rationalism, a mindset that sought to understand the world in technical terms born out of the industrial age. Everything was thought to be measurable and therefore also calculated. While it might seem obvious today that this will not always be a valid approach, it is how it was done. Still today, the idea that all human problems can be solved using technology that exists or is about to exist is not uncommon [36]. Every problem was dealt with as if being a so-called "tame" problem, which refers an easily definable problem with a simple solution. In the light of the dissatisfaction expressed by the social movements of the time, there was a need to develop a more sensitive approach to handling society's issues in a deeper and more sustainable manner. Two professors, Rittel and Webber, coined the term "wicked" problem, characterized by being the opposite to a "tame" problem. A mathematical equation is a "tame" problem in the sense that it has a definitive solution and that it can reach a state where it is considered solved. That is one end of the "tame"-"wicked" problem spectrum. On the other end we have problems such as how to eliminate poverty in a society or how to design a solution to a decrease in mental health among young people. Rittel and Webber argue that such problems can at best temporarily be solved, but that the problem space constantly changes, and grows out of the solutions. Working with "wicked" problems is therefore a forever ongoing endeavour, that should always be re-imagined within a problem space that is never fully understood. To provide a more detailed definition of the concept, Rittel and Webber presented a list of characteristics that are common properties of "wicked" problems. They are listed below, briefly described and how they relate to this thesis.

The statements made are in relationship to the problem of driver monitoring, which can not be formulated. An attempt to put it into words would look something similar to "how do we design the sensing and feedback of an advanced driver monitoring

system to reach a high enough performance necessary to reach the vision zero of deadly traffic accidents". The reasons why this might be the least vague description available is described below.

There is no definitive formulation of a "wicked" problem. A "tame" problem can be described in an complete way that encompasses the entire problem space and provides the receiver with sufficient information to solve it. This is not possible for a "wicked" problem, due to it residing in a unknown domain, or an ever-changing one. In the context of this thesis, it is clear that driver monitoring does not have a clear definition in the sense that neither legislation nor recommendations based on science and statistics are able to formulate a finite problem space. It is partly understood and accepted, the driver must be monitored until full automation is reached, but how to do this and what expressions and features to monitor is not close to being agreed upon.

"Wicked" problems have no stopping rule. To reuse the example from mathematics, the process of solving an equation ends when x is found. The answer lies within the borders of the problem space. This is not the case with driver monitoring. There is no absolute definition of how a perfectly accurate monitoring performance would look. It is of course possible to imagine a driver monitoring system that reaches 100% accuracy on the classes that the developers provide. But understanding what those classifications should be, and in the extension, how to react to them, is beyond the grasp of the state-of-the-art models of today.

Solutions to "wicked" problems are not true-or-false, but good-or-bad. The mathematical equation yields a solution that can be controlled using the well defined rule-set of mathematics. The solution can inherently only be in a state of true or false. In the domain of driver monitoring the different performance states will never exist in the binary, and can at best be understood as a relative performance in comparison to other driver monitoring systems. As mentioned above, a classification model can reach 100% performance on the classes that the developer provides, which is to create a finite problem space under the umbrella of driver monitoring. But it does not solve the general problem of driver monitoring.

There is no immediate and no ultimate test of a solution to a wicked problem. Similar to the truth state of a solution, it is not possible to test a proposed solution to a "wicked" problem in a general way. While it might be possible to test the performance of a unique implementation within a constrained environment, the results of the test can not be trusted to generalize well.

Every solution to a "wicked" problem is a "one-shot operation"; because there is no opportunity to learn by trial-and-error, every attempt counts significantly. In the sense that it is possible to improve the performance of the driver monitoring models, it is possible to incorporate an iterative trial-and-error approach to the development. As mentioned previously, this is not the actual problem but rather "tamed" branches of the main problem, which in turn is not possible to explore by trial-and-error.

"Wicked" problems do not have an enumerable (or an exhaustively de-

scribable) set of potential solutions, nor is there a well-described set of permissible operations that may be incorporated into the plan. Linking back to the mathematics problem, we must be able to reach definitive solutions in order to count them.

Every "wicked" problem is essentially unique. Some parts of a "wicked" problem may be comparable to similar instances of the problem. In the case of driver monitoring, we can compare it to other scenarios where machine learning models are taught to classify different human actions by analysing the relationship between the frames in an image file. In that perspective, a solution to the problem is not unique. What makes it unique is rather the problem space, and the sum of its undefinable properties.

Every "wicked" problem can be considered to be a symptom of another problem. "Wicked" problems can to some extent be defined by their relationship to other problems. In the context of driver monitoring, and its challenges, there are many lesser problems, component problems, that can be identified. For example, to discerning the specific intention behind the driver's actions or behaviour, the system must be able to understand the underlying reasoning of human beings. This type of information is not necessarily readily available even for the driver that is performing the action, since a majority of human behaviours are automatic reflexes. The need to understand the component problem in this case becomes evident when attempting to design advanced warning systems to assist the driver in focusing on the task of driving. Furthermore, solving the underlying problems of a "wicked" problem does not solve the "wicked" problem itself.

The existence of a discrepancy representing a "wicked" problem can be explained in numerous ways. The choice of explanation determines the nature of the problem's resolution. Referring back to the example problem of how we design a monitoring system with sufficient performance to eliminate driving accidents. As this is a proposed problem statement, we must assume that it is just one way to understand the problem from one of an infinite amount of possible angles. In order to attempt to contain it in a sentence, we must delimit it, and delimiting it "tames" its "wickedness" while simultaneously limiting the potential impact of its solution.

The planner has no right to be wrong. Similar to the success measurement of a "wicked" problem's solution being either good or bad, it is not possible to be right or wrong when choosing the approach to a "wicked" problem.

3.2 Human centered AI

The concept human centered AI stems from the idea that AI has the potential to increase the well being of humanity provided that it is designed in such a way. To assure this, the design of AI powered systems must take human values, individual goals and design aspirations into account [37].

3.2.1 Human values

Regarding human values, a common fear is that an expansion of AI systems will result in unemployment and a decreased sense of being valuable for many human beings. While this is a possibility, it is in the designers control to build AI powered systems that are not taking power from the user, either as a conscious delegation or simply without asking, but rather extends the power and ability of the user [37, p. 3]. It is important to note that the risk of negatively affecting rights, justice and dignity is not unique for AI, but rather new technology in general.

3.2.2 Individual goals

From the individual human's perspective, AI has already started to impact us on a self-efficacy, creativity, responsibility and social connections level. Self-efficacy relates to whether or not work that is being done by humans with the assistance of AI removes the ownership of the creation from the human. This becomes very evident in the field of art and creativity [37, p. 61], where the AI models not only extend the ability of the user, but may also adds their own creative touch. One can imagine how a well designed brush tool in Adobe Photoshop may reduce shortcomings in the users painting skills, based on a carefully designed settings GUI under full control of the human. In this case, the creative decisions are not handed over to the AI, and the creation may still be considered a product of the user's mind. Taking it a step further, into the domain of generative models, the lines begin to blur. If a work of art is created by the AI based on the users prompt, the artist may start questioning the self-efficacy in their artistic endeavour. The structure of the example holds true in high-risk environments, where the question of artistic ownership is replaced by responsibility over unexpected consequences and a potential for accidents. The complexity of self-efficacy and responsibility becomes particularly apparent in self-driving cars, as the autonomy of our vehicles are increasing [37, p. 66].

3.2.3 Design aspirations

The third topic of human-centered AI is design aspirations, which refers to the practice of designing reliable, safe and trustworthy use-cases of AI. Before we dare to delegate our sensitive tasks to the machines, it must not only reach the appropriate level of performance, but also exhibit it in order to evoke trust [37, p. 9]. In extreme cases, such as with self-driving cars, we will not be satisfied with a 99% chance of success [37, p. 45]. A central idea in human-centered AI is that high levels of automation does not necessarily mean low levels of human control, provided that the system is carefully designed to be an extension of the user and not a separate autonomous entity. While driving a semi-automated vehicle, the driver will in periods delegate the control of the vehicle to the system, and take it back when it is necessary. Due to external events, there will also be situations during which the system must ask the driver to take control. The interface through which the driver and the system communicates is highly complex in its multi-modality, and the possible ways that the system may be designed to communicate with the driver

is numerous. This puts very high requirements on the design in order to make the interaction seamless.

3.3 Human-AI interaction

Human-AI interaction may be considered a branch of Human-Computer interaction (HCI). While the interaction often is similar to classical HCI, AI-infused systems come with a variety of novel opportunities and design challenges. AI infused systems, i.e. systems that feature AI capabilities directly exposed to the user are often poorly understood due to probabilistic behavior that is often not possible to fully explain. This is particularly present in large language models that produce output based on a large amount of fine details which the user is not aware of. An intelligent car system can also be considered an AI-infused system, in particular the ADDW part of the system that interacts with the user based on what it is sensing through its multi modal sensors.

To better design AI-infused systems, Amershi et al. [38] developed, evaluated and proposed a collection of AI design guidelines in 2019. The guidelines come in four categories, relating to which part of the interaction the guidelines concern. **Initially** it should be very clear what the system is capable of, and how reliable it is in those tasks. The limitations of an AI system should always be very clear, and this should therefore permeate the interaction from start to finish. **During the interaction** the system should be very mindful with its timing when it comes to potentially interrupting the user. The unprompted output from the system should balance between not being strong enough, and not being intrusive. The interaction should also take into account the user needs and current environment, as well as their social and cultural context. Furthermore, the system should not reinforce undesirable and unfair stereotypes and biases. **When wrong**, there should be several ways to either correct the system or disregard its services. The AI system should not in any way be standing in the way of the task of the user, but rather be a background assistant that provides help from the user when requested. Transparency is also a cornerstone, and it should be clear to the user why the system made a mistake in order turn it off or correct it in an appropriate manner. **Over time** the system should use the interactions that have been made to adapt to the specific needs of the current user, and use this insight to tailor the interaction. It is important that this development is transparent to the user, and to allow the user to customize the adaptation.

4

Methodology

This thesis is on the highest level the development of a machine learning model using a hybrid approach of classical machine learning methodology and design methodology. This model, which detects secondary tasks while driving, is intended to be used as a material for designers to subtly inform the driver to stay focused while driving.

4.1 Research *for* design

The research and insights that will be the product of this project may be considered *research for design*[39]. Frayling, one of the prominent researchers that legitimised design as an academic research field, differentiates design research into three approaches: Research *into* Design, Research *through* Design, and Research *for* Design [39]. Briefly summarised, Research into Design is the study of the field of design itself, with designers and design as the research subject. In Research through Design, the designing of artifacts is used a methodology for generating new insights that emerge from the design process and the designed artifact. Research for Design is about doing research that prepares and informs future design work. Frayling describes this as the gathering of reference material. This form of design research is more commonly found in industry, including activities such as gathering insights about target users and bench-marking competing products. While Research through Design has gained traction in the academic design research community [40], [41], Research for Design has seen less engagement. While the development of a driver-monitoring system could be considered a Research through Design activity, the main purpose of the thesis is of a Research for Design nature; the answer to the research questions will be of use for the designer that is to develop the interface between the driver and the machine. This will be further elaborated on in the discussion section of this report. In essence, the work that that this project will be composed of is based on the idea that the design choices that are made, are done with the final advanced driver warning system in mind.

4.2 Double diamond

The double diamond framework [42] was designed by the British Design Council as a tool to aid design processes. In essence, it provides an overview and a road map over the design process by defining and separating it into distinct phases. While a design

process seldom is linear, it is widely accepted that the four phases are commonly occurring in some shape or form. The silver lining is that every phase can be understood as either diverging or converging, that the process is either broadening the scope or narrowing down.

Discover An appropriate initial approach to solving a problem is to understand it in its own domain. Depending on the nature of the problem, the phase of discovery might look very different. Building understanding might entail interviewing stakeholders or gathering information available in the literature. This is a phase of divergence, and the goal is to draw the big picture by accumulating as much relevant data as possible.

Define Using the data collected in the discovery phase as a foundation, the defined phase consists of building insights from the data. As the complexities and nuances of the problem are better understood it is possible to decide on an approach and to narrowing in on a well defined problem space.

Develop With a clear view of the problem space, it is time to start developing. During the development phase testable prototypes emerge and are explored. Depending on the project and design goal this can be anything from software to physical objects. Being a diverging phase, exploration is encouraged, and numerous reiterations are common. Questions are answered through testing and new questions emerge. Practical insights can present the need to go back to previous phases.

Deliver The line between the development phase and the deliver phase is not always clear, since there might be some going back and forth, striving towards a final product or solution. The delivery phase, even though it is a converging phase, often involves a lot of exploration and reiteration.

4.3 A generic model of machine learning methodology

Machine learning is a branch of artificial intelligence that involves automating computer tasks like classification and decision making. It is about learning from data and finding patterns invisible to the human eye. One accepted definition is posed by Tom Mitchell and follows: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E [43]. It is in other words about machines, i.e. computers, learning from experience which is represented as data. The steps in developing a machine learning model vary from use-case to use-case, but the most generic approach includes six steps. [44].

Collection and Preparation of Data In essence, a machine learning model is molded from data that is describing the relevant contact points of the environment in which it is to operate. Depending on the use-case, the data is either readily available or needs to be collected. Data collection might, for example, be done by the use of instruments like video cameras or by scraping the internet of vast amounts of natural language. Raw data is often noisy and unstructured, and must in that

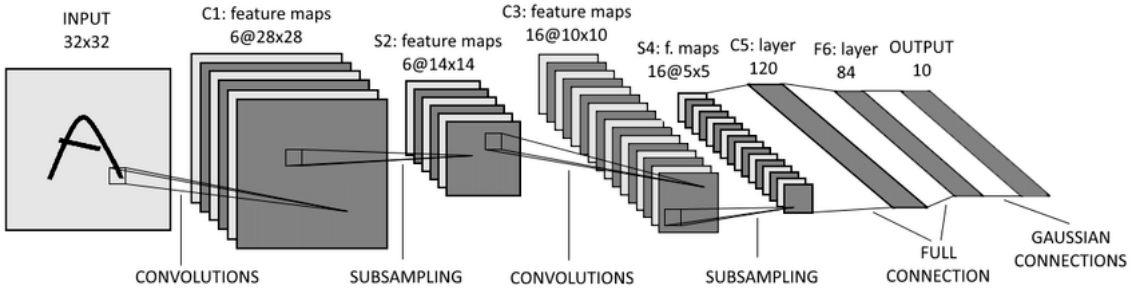

Figure 4.1: LeNet [27]. A model's architecture may be described by its different layers and filters.

case be cleaned. The final product of the data preparation phase is a clean and well understood dataset.

Feature Selection The features, i.e. patterns, in the data are what the model learns to differentiate between. This could mean the different relationships between the pixels in an image of a cat and a dog [45], but also more abstract patterns like variations in radio waves [46]. It is important to note that the features are what guides the decision of a model, it is not something that is inherent in the data itself, but rather in the eye of the observer. Although it is not always possible to predict what features will be the most crucial for the final model, it is often possible to get a general idea by exploring and developing an understanding of the data through visualization or descriptive statistics.

Choice of Algorithm There is seldom an obvious choice of type of algorithm, and a substantial amount of work done in the machine learning community involves finding new use-cases for well-established algorithms. Based on domain, amount of data available and computational resources, the pros and cons of different algorithms can be weighed in order to choose the most suitable one for the current context. It is not uncommon to develop and train several types of algorithms, in order to compare their performance.

4.3.1 Selection of Models and Parameters

While it might be possible to find finished model architectures available online, it is often necessary to tailor the model in a way that suits the specific task at hand. As with the choice of algorithm, the best parameter settings are not always obvious and approaching the problem by training different models for comparison is a common approach.

Training When the architecture is built and the parameters are set, the models can start to take form by learning through processing of the dataset from earlier. The dataset is often separated into a training dataset and an evaluation dataset. The training dataset is learned by the model, and the evaluation dataset is used to assess whether or not the model performs well on new data. It is worth mentioning that a well performing model might not perform as expected "in the wild" after being deployed. If both the training and evaluation data come from the same source, and

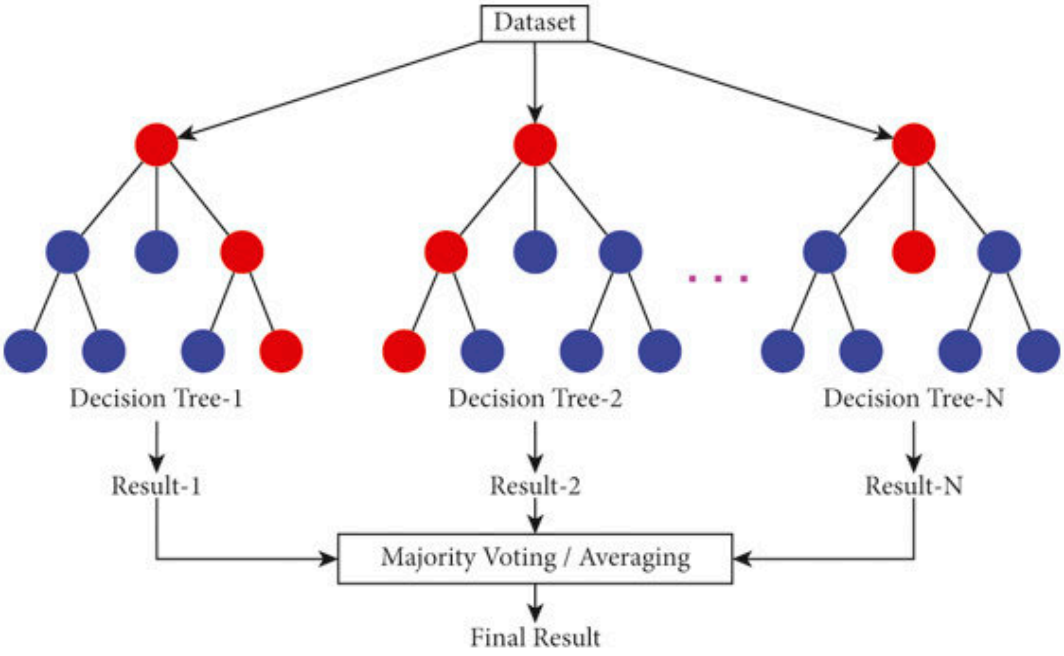

Figure 4.2: Illustration of the random forest architecture.

the source does not represent the real world, the model might find itself unfit for the task at hand.

Performance Evaluation The training phase and evaluation phase often overlap, and there is often some going back and forth between the last phases of the development. Some parameters can need adjustment, and some exploratory models might prove to have unforeseen issues. The model is tested on the evaluation dataset, and performance metrics such as accuracy, precision and recall are measured.

Potential model architectures The choice of architecture will be based on the amount of training data available. Since all the data used for the modelling will be collected, this quantity is expected to be relatively low which excludes deep learning methods. The task for the models to solve is classification, which could potentially be done using SVM, HMM or some form of ensemble method such as adaboost, XGBoost or random forest. Due to previous experience and knowledge available in the project, a random forest algorithm will likely be the most suitable option.

Random forest The random forest algorithm was introduced by Leo Breiman and Adele Cutler in 2001 and is an ensemble machine learning method that can solve regression and classification problems. The name comes from its architecture being composed of N decision trees, nodes, each responsible for a random subset of the features in the input vector. When making a prediction, each tree makes a prediction based on the features it has learned, and the final prediction is based on a majority vote of all trees.

4.4 A hybrid approach

This thesis will make use of a hybrid methodology by combining the double diamond design framework and the generic machine learning development workflow.

Discover The work starts by exploration of the domain, through a literature review and by talking to people involved with development of the different driver monitoring systems at the company Smarteye. Being mindful as not to make any early decision, and not constraining the literature search to the domain of driver monitoring or cars in general. Machine learning models are not domain specific, and the thesis can be expected to draw inspiration from other domains like, for example, general human behaviour and action recognition. The outcome of the discovery phase is a collection of insights from a wide variety of literature, including academic publications and text books.

Define The gathered information lays out the background of the work to come, defining concepts that will occur throughout the thesis. Decisions regarding the data collection will be made in detail. This includes how the data collection environment will be set up, what script the participants will follow and how the data will be saved.

Develop The data will then be collected and recorded using staff members from Smarteye and the aim is to get approximately 40 people. Each session will take 10-15 minutes, and the participant will be driving in a simulator while being prompted to do several tasks while staying on the road. While going through the tasks, the data will be collected using a double camera setup that records and processes the data. Since the DMS and CMS models are already existing, those will simultaneously be making classifications and outputted as a csv file. The data will then be pre-processed and ordered in such a way that it can be used for the modelling using outputs from both the DMS and CMS. Furthermore, the data will be separated into a training and evaluation dataset.

At least two architectures will be chosen for training, and the training will be done on powerful computers owned by Smarteye that is located in their office building.

Evaluate After the training, the architectures will be tested on measurements such as accuracy, precision and recall. The development and evaluation phase will likely be iterated several times to attempt to increase performance.

5

Procedure

5.1 Defining phone use

In order to proceed, the complex and broad action class of phone usage had to be defined. Human behavior may be easily recognizable to the human eye, but in order to make it identifiable to the machine very strict rules have to be defined. The main behavior class that this project is focusing on is phone usage while driving. The approach was to base the definitions on the literature in combination with interviews of research engineers and technical developers at Smarteye over several iterations until the direct stakeholders at Smarteye were content with the level of detail. The methodology follows a double diamond design structure. Defining the scope, producing definitions, defining what the data collection will cover, producing task instructions.

An initial discussion was held to decide on the largest and most high level classes to be covered by the definitions. The goal was not to create something that might resemble a definition, but rather points for further discussion as to not make any early decisions and thus limiting the scope. A secondary task event is defined as an event that is sandwiched in between periods of natural non-distracted driving. In the context of driving, the actions that go into controlling the car are considered the main tasks. Therefore, a secondary driving task refers to actions that separates the driver's attention from the main task of driving.

Main class	Umbrella definition
Talking on phone	Driver talking on the phone
Interacting with phone	Driver interacting with the phone
Other	Driver behavior similar to phone usage without a phone

The main takeaways and further questions from the discussion are listed below:

- Human behavior and actions is comprised by series of discrete actions
- Talking and interacting may overlap and can thus sometimes not be separated
- Other behavior might include a phone and should only be defined as other behavior if the phone is not talked through or interacted with
- Can talking and interacting be separated?

- How may interacting be defined?
- Tasks should be defined in terms understandable by the machine
- The modelling in the project is constrained by the amount of data that will be available

Using a general example of making a phone call, it is clear that talking and interacting are commonly part of the same action chain that makes up a unique phone usage event. The driver takes the phone in their hand, interacts with it and places the phone close to their ear and talks. The question is then whether or not it is meaningful to attempt to separate those actions into smaller components and define those separately. One secondary task event is, as mentioned, a series of actions that is to some extent solving a problem or changing the state of the interior or exterior environment. The actions, for example lifting an object or twisting the cap off of a bottle, are by themselves not meaningful even though being essential for the entire task. While a task like drinking from a bottle may be varied in many different ways, being made up of many different component actions such as extending a hand or grabbing the bottle, some of them are likely not possible to skip. For example, to drink from a bottle the bottle must be opened which may be considered an essential component of the behavior of drinking from a bottle. This can be applied to all secondary driving tasks as long as the level of definitions are sufficient.

5.2 From definitions to secondary tasks

The next step was to come up with a first draft of suggested secondary tasks and define them based on conditions met at some point during the tasks. The motivation behind the tasks chosen was based on the level of risk associated with the secondary tasks [47] and the corresponding definitions.

Table 5.1: Secondary tasks first draft

Secondary task	Object in hand	Hand near head	Looking at object	Talking
Phone to ear talking	Yes	Yes	No	Yes
Phone in hand talking	Yes	No	No	Yes
Phone on leg talking	No	No	No	Yes
Talking to passenger	No	No	No	Yes
Phone in hand interacting	Yes	No	Yes	No
Phone on leg interacting	No	No	Yes	No
Reading on phone	Yes	No	Yes	No
Eating an object	Yes	Yes	-	No
Drinking from object	Yes	Yes	-	No
Checking notification	No	No	-	No

While tasks like *Phone on leg interacting* likely includes **Object in hand** at some point, it is not something that the machine must learn in order to recognise the task. During the time which the secondary task *Phone on leg talking* is taking place, the conditions in the table above are true. **Looking at object** will not be true during

all the moments of time that is composing the secondary task, but often enough to be considered an essential condition. This is where the dimension of times comes in. Interacting with an object, if the interaction is a secondary task, will recurrently be "paused" in order to return the attention to the road and the task of driving. There will therefore be times when the driver is not distracted and focusing on the main task of driving while holding the phone in their hand, and momentarily shifts their gaze to their phone to continue to write a text message or look at a map. A *Phone in hand interacting* secondary task event must therefore be considered the entire time span starting from the driver grabbing their phone to when it is put back down.

5.3 The final list of secondary tasks

Based on the first draft of secondary tasks within the scope of this project and their definitions, a workshop meeting with technical experts on Smarteye was held as to decide on which tasks to include.The main factors behind the following decision was that there was a need to include as many variations as possible for tasks including talking on phone, interacting with phone and other behavior that would potentially easily confused with phone usage. Furthermore, the future requirements from [3] and [4] were taken into account. The following table is the outcome of the workshop.

Table 5.2: Secondary tasks

Task name
Natural calling and talking on phone
Natural calling and talking on speaker phone
Natural texting (interacting) on phone
Natural reading on phone
Phone in hand interacting
Phone on leg interacting
Phone in hand to ear talking
Phone in hand talking speaker
Phone right hand on leg talking
Phone in hand pinned ear to shoulder talking
Talking to passenger
Phone in fixed position interacting
Phone in fixed position talking
Remove phone fixed position
Infotainment interaction
Reading on phone in hand
Eating an apple
Drinking from bottle
Reaching for object passenger seat
Checking notification phone

5.4 Expert interviews

After the definitions had been made, the next step was to have interviews with experts at Smarteye that had experience with modelling secondary tasks and the tracking of secondary tasks related to phone usage. The interviews surrounded the two sub questions to research question 1.

RQ1-a: *Which sensory signals can be important in identifying and classifying phone usage in the context of driving?*

RQ1-b: *Which phone-related secondary tasks are most prone to produce false negatives?*

5.4.1 RQ1-a

Regarding raw sensory signals, both the CMS and the DMS use IR cameras to capture image data, which is fed into different models to extract features. Both systems use object detection algorithms to identify and track objects in the images. This can be considered to create signals which are based on raw sensory data. Some of the more useful ones are body key points in 2D and 3D, eye tracking and object detection.

5.4.2 RQ1-b

The CMS and the DMS are developed for different tasks, even though they behave similarly. Both are sensitive to both occlusions and strong light sources like the sun. Therefore, the hardest phone related secondary tasks that are most prone to false negatives are those where the phone may get occluded. For example, during phone to ear events and similar.

5.4.3 Preparing data collection

The next step after defining which secondary tasks to include was to start designing the data collection. While being similar to a experimental setup, there were no independent variables to balance. In essence, the goal was to create a setup that were in some ways immersive enough for the participant to behave as if they were driving in a real car. when it comes to shifting their attention between the task of driving and the secondary task.

Physical simulator For the physical simulator setup, Smarteye provided a set of seats that had been used for demonstration on a previous exhibition together with a Logitech G920 steering wheel, pedals and clutch setup (See Figure 5.1). In order to attach the steering wheel and the recording cameras metal railings were used to build a stable rack. It was essential that the cameras were set firmly in place, since it would otherwise not allow for precision on the spatial data. The position of the cameras were later controlled in the calibration of the recording software, to assert that placement and angles were correct. Furthermore, a large tv screen was bought to behave as a windshield for the simulator environment.

Figure 5.1: Simulator setup

Simulator environment To provide a simulator environment, i.e. the "game", several options were explored. What was sought after was not a perfectly valid simulated driving environment, but rather a driving-like task that would require the participant to keep a similar level of attention as when driving a regular car in traffic. The environment Carla was first tested, which turned out to be unsuitable due to difficulties in setup and a bad physics engine. This was expected, as Carla is geared towards training automated driving models. The next candidate was to develop an environment in Unity, and to build it from assets bought from unity asset store. While being a promising option, as it allows for full control over the environment, it would have been too time consuming. The last option was to explore readily made environments and even games to see what was available. After some searching, two games were found, BeamNG and Euro Truck Simulator 2 (ETS2). After testing out free demonstrations of both games, BeamNG was excluded due to being a racing game. ETS2 on the other hand turned out to be the most suitable thanks to being designed towards a natural driving experience. The concept of the game is that the player is a truck driver and is building a business from scratch. Furthermore, it is an open world game that allows the player to explore a downscaled version of Europe using a truck. To find a suitable driving scenario in this simulator world, a few hours were spent driving around and testing out different routes. A route starting in Rotterdam heading towards Frankfurt via Amsterdam was chosen, due to being mostly highways and therefore a suitable driving context for secondary tasks. To provide a more natural view, a bumper camera was used to place the camera view closer to the road and thus more closely resembling the windshield view from a regular car.

Data recording system The next part of the setup was the data recording, which were to be done using a CMS and DMS camera connected to a Linux computer

running Smarteyes product Trackergui, which is a video recording software with an ensemble of signal processing algorithms that is generating logs of all signals. As previously mentioned, the cameras were attached to the metal rack surrounding the simulator cabin.

Soundboard to play instructions The core of the data collection was for the participant to perform secondary tasks while driving. The instructions, which were to be based on the secondary tasks mentioned above, had to be delivered in an easily digested and clear way. The initial idea was to read the instructions out loud while the participant was driving the simulator and was recorded. This was soon deemed not possible, as the effort of the person in charge of the recording would be too high in combination with overseeing both the recordings and simulator. To solve both this issue and to assert that the instructions were consistent for all participants, a decision was made to create sound files from the instructions using a text-to-speech (TTS) model. Initially the free python library GTTS was tested, but decided to not be used due to the voice sometimes not being perfectly clear. After searching for alternatives, OpenAIs text-to-speech model was used with the voice called Shimmer. The model was access through the Python API and instructions were generated based on the secondary tasks that were to be recorded. The first idea was to use an audio playing program and to create a playlist that would run as the participants were driving and give instructions to perform the tasks at balanced intervals. This turned out to not be a very robust solution, since it did not give the recording leader very much control over the playback and would easily break the flow of the recording if anything unforeseen happened. Every different type of task, for example tasks including talking, was set to a fixed duration to generate sufficient amounts of data. If the participant crashed in the simulator environment, the playback was paused until the participant was naturally driving again.

To increase the control over the flow of instructions, and allow for pausing the playback when necessary, a soundboard interface was developed in python using tkinter and pygame. This also made it possible to save UNIX timestamps whenever an instruction was delivered, which would later help immensely during the annotation and labeling of the data 5.4. Every secondary task scenario starts with an instruction describing with as few words as possible what the participant should do, and ends with the participant being asked to return to a neutral driving position which is described as how the participant would sit and behave in a comfortable way while driving naturally.

5.4.4 A data collection session

The sessions started with an introduction to the simulator setup, by letting the participant sit down in the driver's seat and driving around in a test scenario to get familiar with the setup and the experience of driving. While doing so, the participant received instructions on what would happen during the following recording session. In order to keep the mental effort of the participant low, the introduction was kept as a casual conversation. The key instructions that were checked off by the recording leader are listed below:

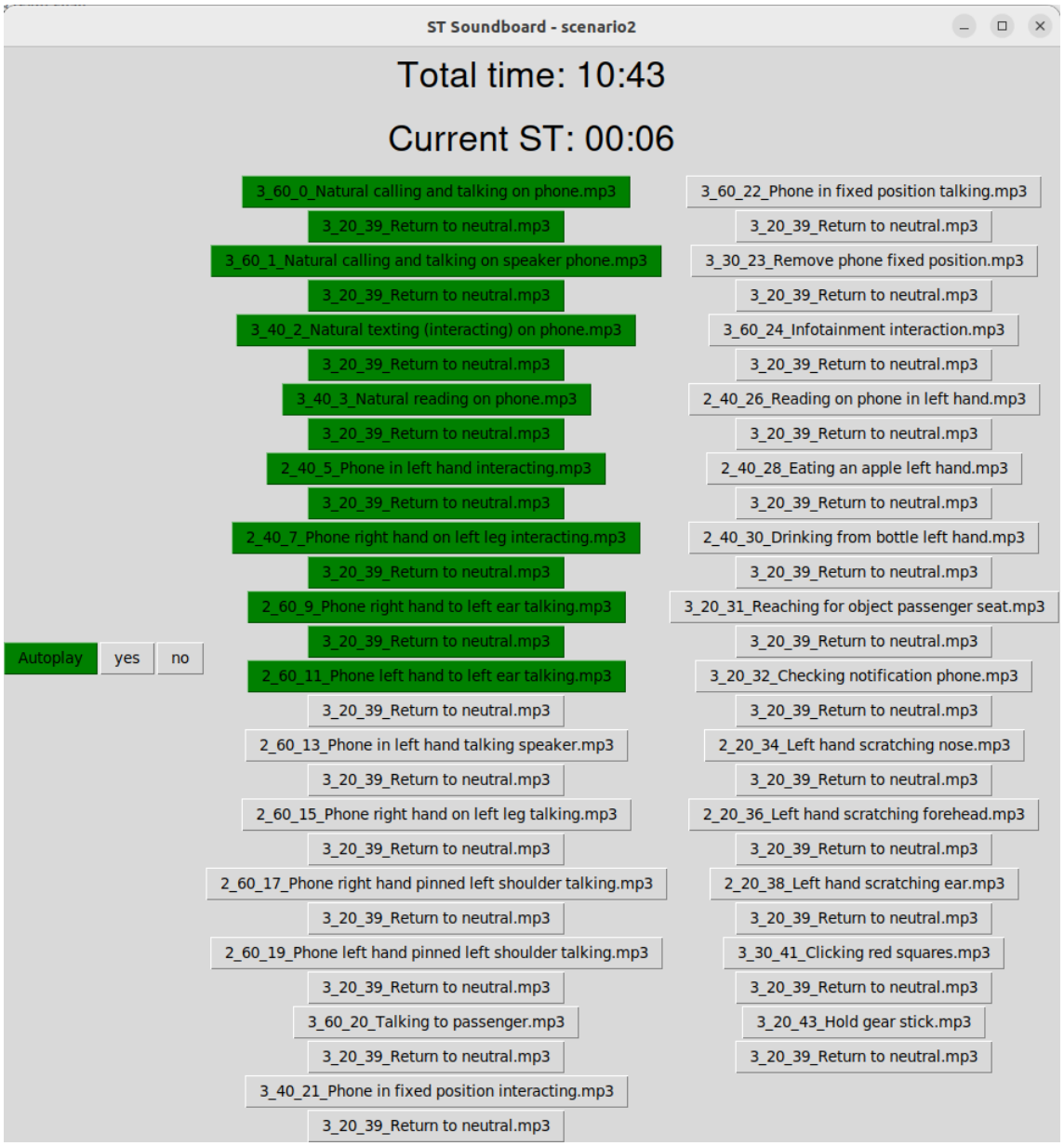

Figure 5.2: Soundboard interface

- *Some tasks that you will receive will ask you to interact with your phone by sending a text message to a friend. You should then actually write a message, whether or not you send it is optional.*
- *Some tasks will ask you to talk, and you have two options. Either you describe what you see in the simulator environment or we play a game of 20 questions.*
- *Some tasks will ask you to read on your phone, and you should then go to a web page in your web browser and read. Please take your phone now and prepare.*

The rules for 20 questions are the following. The recording leader is thinking of something, and the participant gets to ask questions to narrow it down to what it is. The targets used were a cat, a stone, a raspberry pi computer and the sensation in the stomach when a roller coaster makes a loop.

The participant then got the opportunity to ask questions before the main driving scenario was loaded into the simulator and the recording was started. The soundboard playback was then started and set to auto play.

5.4.5 Piloting

Piloting was done in several iterations with volunteers in the research department of Smarteye, fine tuning both the secondary task instructions and the introductory instructions. The main takeaways from the three piloting sessions were the following.

- Examples of the different secondary task instructions should be given beforehand, since the instructions might be confusing while driving.
- The participant should get to test the phone holder beforehand, since it is very hard to figure out how a phone holder works while driving.
- The pedals should be fastened.
- The participant should be asked to use the break during the test scenario, in order to get a feel for how long the braking distance is.

5.4.6 The data

One data collection session, with one participant, generated two separate recordings. One recording from the CMS camera and one recording from the DMS camera. The format that trackergui is using by default is .sec, which is essentially a folder with a series of .raw files containing video data with very high resolution, but also other metadata such as timestamps from the recording. The precision is using a microsecond accuracy and is provided by the clock frequency of the computer running the recording software. Furthermore, it is also producing log files, which are large .csv files, containing one row per frame in the video. The columns are the different signals, some heavily filtered and others raw data, and can be identified by their timestamps. In order to be able to connect one frame in the video recording

with the log files, the timestamps are used. This also allows for the possibility to line up the CMS and the DMS recordings, which would be very useful later on.

5.4.7 Annotation

Annotation, also called labelling, is the process of labelling the data with the outcome that the machine learning model will learn to predict. In this project the labelling was done by stepping through the video recordings frame by frame, and setting different event flags as active or inactive. One example of this is to mark every frame of the video with whether or not the participant has a phone in their hand. This is then used as ground truth for the training. More on this below 5.4.7.

Choosing the tool Before starting the labelling work a number of software options were explored. The recording tool iMotions was first tested. iMotions is a tool that allows the researcher to build and run experiments while recording data from different sensors such as cameras, eye trackers and heart rate sensors. Since iMotions also allows the user to label the recorded data, the idea was to load the recordings from the trackergui software in iMotions and then label it as if it would have been recorded in iMotions. Several issues came up trying to do this. First of all, the file format on the recordings from trackergui was as mentioned above .sec and iMotions only accepts .mp4 and .avi. To resolve this, the .sec files had to be converted into .mp4. Due to varying frame rate in the trackergui recording, the method to do this was less than straight forward. The issue was that when converting to another format was a loss of the meta data mentioned previously. The .sec file contains information on when the recording was made, i.e. the exact timestamps with microsecond precision. This means that when looking at the video and noting the exact frame when an event starts, this will align with the log files and it is possible to tell which row in the .csv files corresponds with the frame in the video. While this might sound trivial, it becomes hugely important when trying to align the CMS log with the DMS log. The loss of metadata in the conversion in combination with iMotions not being designed for the use-case of importing external recordings, excluded it as a potential labelling software. The next software explored was Darwin from V7 Labs. This was excluded early due to not supporting the necessary file formats. The options were therefore limited, and the idea to develop a custom annotation software using python and pyqt5 was proposed. After laying out the plan and describing the program, the idea was abandoned due to the amount of working hours necessary to build something useful. From conversations with people working at Smarteye who have had similar issues before, a rumor of an in-house labelling software emerged. After doing some more research and inquiries, this turned out to be true. A software existed that would take a .sec file as input and allow the user to create labels based on the timestamps from the log files. Being designed for the task at hand, it was decided that this was the optimal solution. This tool was called *Event annotation tool*.

Labelling rules Before starting the labelling work, labelling rules had to be carefully described and agreed upon. This is usually absolutely necessary when more than one person is labelling the same data set, since the variation might be great if

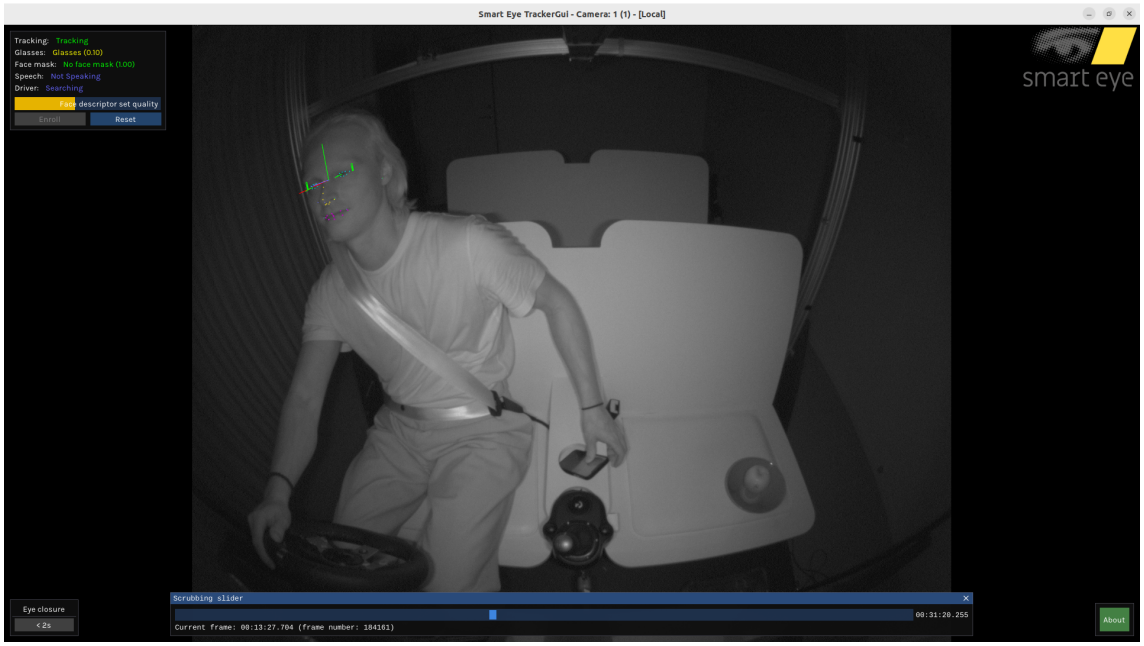

Figure 5.3: Replaying a recording with trackergui, see slider for frames and timestamp

left to interpretation. The different label classes were based on both the essential components described above and by carefully observing the recordings. The sensitivity is 5-10 frames, which for example means that an *ObjectInHand* event starts when the participant has grabbed an object in a static position and the object has moved for 5-10 frames. The event ends when the object has laid still for 5-10 frames after being released. As mentioned previously, this becomes less obvious with interaction events. Often the participant initialises interaction with the phone and then pauses the interaction for several seconds to focus on the driving. In cases like this, the time in between direct and obvious interactions are still labelled as *interacting with phone*. The interaction does not end until the interaction is actually over for the current event, which is possible to identify by jumping back and forth in the recordings playback.

As seen in figure 5.4, events commonly overlap. *Object in hand* and *phone use dynamic* always overlaps, but there are also occasions when *object in hand* is active and *phone use dynamic* is not, for example while holding an apple. The reasoning for the overly detailed labelling was to be able to differentiate between secondary tasks that contain many similar essential components, but are due to the final outcome of the task belonging to different classes. In order to be able to train a model on the many classes, the level of detail had to be sufficient.

To increase the speed of the labelling work the timestamps saved from whenever the secondary task soundboard played an event was preprocessed and loaded into the event annotation tool. The data was structured in the same format as how the event annotation tool saves its labels, which resulted in the start (instruction audio played) and the stop (instruction to return to neutral) of the secondary tasks was marked in the timeline. This allowed the labelling to be done more efficiently as less

Table 5.3: Labelling rules

Label class	Start rule	End rule
ObjectInHand	Object is in hand and have started moving from static position	Object have stopped moving in static position
InteractingPhone	Participants is interacting with phone using gaze vector and fingers	Participants have stopped interacting with phone
InteractingInfotainment	Participants is interacting with infotainment system using gaze vector and fingers	Participants have stopped interacting with phone
Talking	Participants mouth is moving	Participants mouth is moving in end of event
PhoneToEar	Phone is reaching a semi-static position close to ear	Phone is starting to move from semi-static position close to ear
PhoneUseDynamic	Phone is in hand and have started moving from static position	Object have stopped moving in static position and is no longer in hand
PhoneUseStaticHigh	Phone is reaching a high static position and stops moving	Phone is starting to move from high static position
PhoneUseStaticLow	Phone is reaching a low static position and stops moving	Phone is starting to move from low static position
Eating	Apple is in hand and have started moving towards mouth	Apple is in hand and have started moving away from mouth in end of event
Drinking	Bottle is in hand and have started moving towards mouth	Bottle is in hand and have started moving away from mouth in end of event
PhoneHolder	Phone is in phone holder and have stopped moving	Phone has left phone holder and have started moving
OtherBehavior	Hand have started moving towards unidentified task	Hand is returning to neutral position

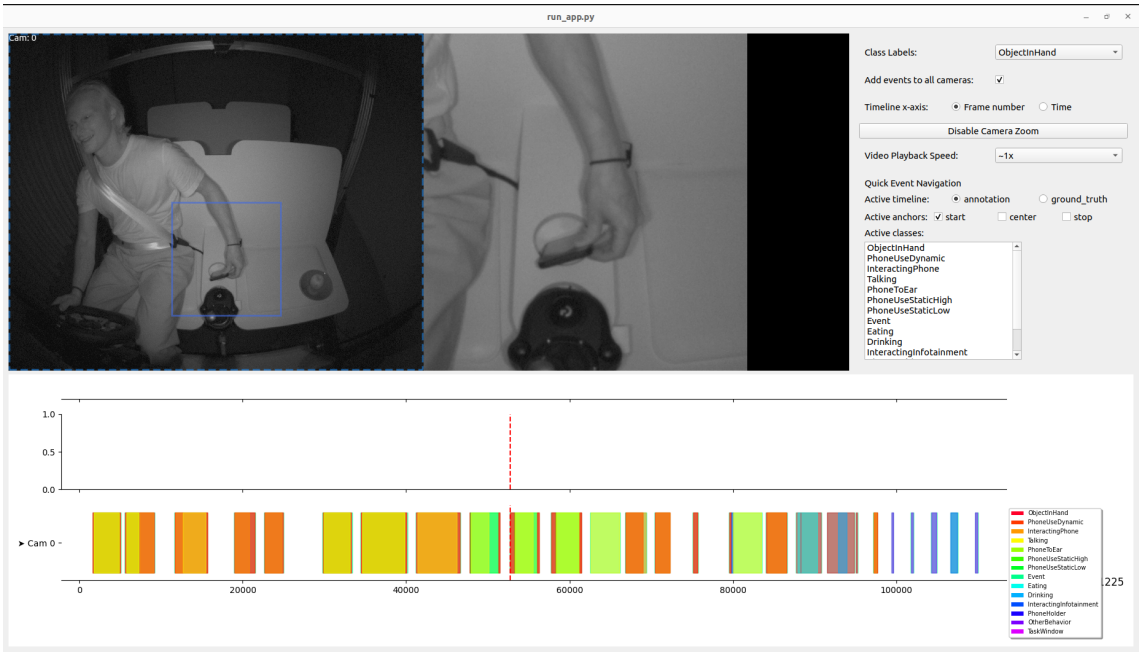

Figure 5.4: Event annotation tool, the multicolored blocks in the bottom row represents discreet secondary task events that in turn consists of many different event classes.

time searching for the start and the stop had to be done.

The workflow of the labelling

- Identify start of secondary task event
- Find and mark start and end of major event classes such as *object in hand*. Major event classes are the most common events.
- Find and mark start and end of minor event classes such as *interacting phone*. Minor event classes are the least common events.
- When every secondary task event is labelled, encapsulate every secondary task event in an *Event* event.
- Save labels to csv.

5.5 Preprocessing the data

After the labelling was finished, the datasets for training the models could be created. The goal here was to create separate .csv files that each contained the frames of one secondary task event for one participant, with both CMS and DMS signals synchronised. The frames also had to be labelled using the label data, which is windows described using frame numbers. The pseudo code is as follows:

The reasoning behind separating the larger logs into smaller files is that the .csv files are quite large, and takes several seconds to open. Storing the data in this form

Algorithm 1 Merge CMS and DMS, Add Labels, and Create CSV Files

1: **Step 1: Merge CMS and DMS**
2: Load CMS and DMS logs as dataframes.
3: Drop unnecessary columns
4: Compare the timestamps of the first row in each dataframe.
5: Remove rows with lower timestamps in the dataframe with the earlier starting timestamp.
6: Align and merge the data frames on the timestamp to create `combined_df`.
7: **Step 2: Add Labels to `combined_df`**
8: Load the label files.
9: Add new columns for each event in the labels to `combined_df`, initialising all values to 0.
10: Iterate over each label event, and set the appropriate window in the corresponding column to 1.
11: **Step 3: Create CSV Files Containing Single Events**
12: Group the combined dataframe by the `"Event"` column.
13: Add padding rows before the event starts in each group.
14: Load soundboard data to determine the order of secondary tasks for the participant.
15: Create and save new CSV files for each event group, naming them according to the soundboard data.

allows for more convenient sorting and quicker debugging iterations later on. It is possible to open a csv using pandas read_csv method() with usecols (USECOLS REF), but it is still necessary to load all the rows.

5.6 Creating additional signals

With the data in order, it was time to get into the data. The logs that were being generated by trackergui contained about 3000 columns of signals. Some examples relevant to the following modelling are viewing direction, object in hand and body keypoints.

5.6.1 Gaze wrist point intersection

As seen in figure 6.5, rays are "shooting" out of the eyes of the driver as normalized vectors, together with a consensus vector based on those called viewing direction. The underlying technology is a computer vision algorithm parsing the image data captured with NIR-cameras. Viewing directions are separated into several columns describing the x, y, z coordinates of the origin of the vector, and a direction vector. Based on the quality of the signal, the system also provides an uncertainty angle which is increasing when the quality of the signal is decreasing. This may be due to a change in light conditions or temporary occlusions. This signal is generated by both the CMS and DMS system, but the DMS is the most reliable. Furthermore, the CMS system also draws body keypoints seen as the white "skeleton" in the same

figure. Those are available both in 2D and 3D and are defined by coordinates.

Using those two signals, together with the object in hand signal, it is possible to create a signal that tells whether or not the driver is looking at an object in their hand. This was thought to help the final models to determine whether or not the driver is interacting with their phone.

This was done by writing an algorithm based on a ray-sphere intersection test from computer graphics. The wrist point from the body key points 3D signals is used as origin for a sphere, the viewing direction origin is used as origin for a ray and the viewing direction direction is used as a direction vector for a ray. The size of the sphere is increased using the uncertainty angle from the viewing direction.

5.6.2 Ray-sphere intersection test

$$\text{distance} = \frac{\mathbf{oc} \cdot \mathbf{d}}{\|\mathbf{d}\|^2} \tag{5.1}$$

The variables used in the equation are explained as follows:

oc The vector from the origin of the ray to the centre of the sphere.

d The direction vector of the ray.

$\mathbf{oc} \cdot \mathbf{d}$ The dot product of the vectors **oc** and **d**.

$\|\mathbf{d}\|$ The norm (length) of the direction vector **d**.

distance The distance from the origin of the ray to the point of intersection with the sphere.

5.6.3 Moving a point in 3d space along a direction vector

As seen in figure 6.5, the object in hand is offset from the wrist point about 10 cm. To improve the precision of the algorithm, the wrist point is offset 10 cm using the corresponding elbow key point, creating a direction vector from elbow towards wrist.

$$\mathbf{b}' = \mathbf{a} + t \cdot \frac{\mathbf{b} - \mathbf{a}}{||\mathbf{b} - \mathbf{a}||} \tag{5.2}$$

The variables used in the equation are explained as follows:

a The initial point vector.

b The target point vector.

t A scalar parameter that determines the position between **a** and **b**.

$\mathbf{b}'$ The resulting point vector after applying the transformation.

$||\mathbf{b} - \mathbf{a}||$ The Euclidean norm (length) of the vector difference $\mathbf{b} - \mathbf{a}$.

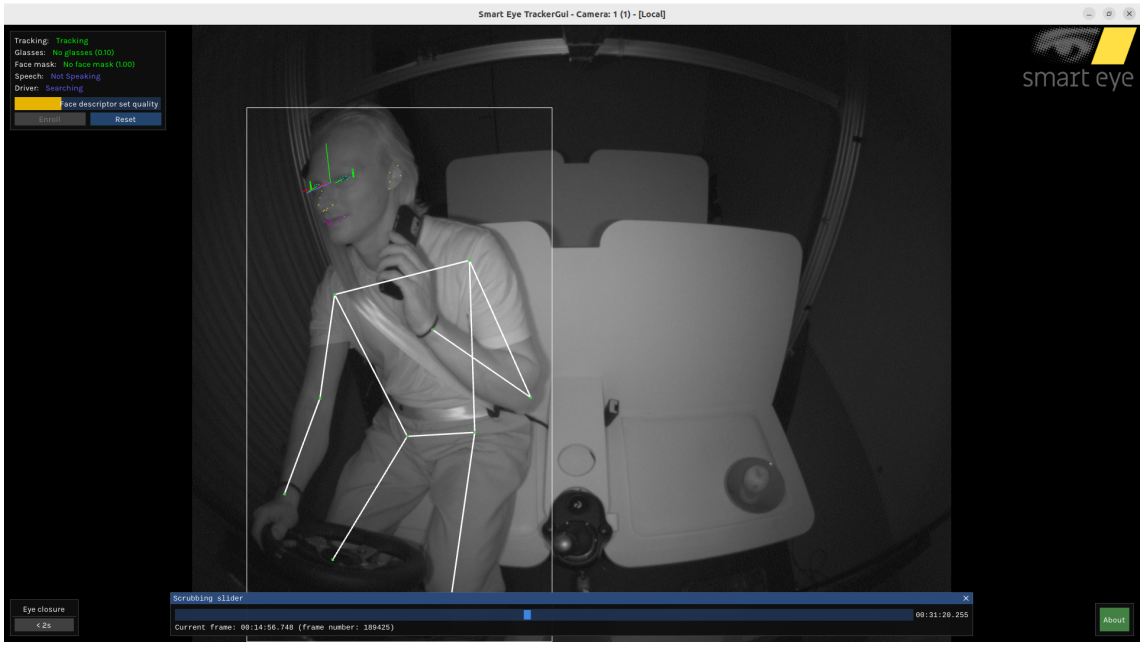

Figure 5.5: Body key points and viewing direction visualised.

5.6.4 Distance between two points in 3D space

Another signal created for this thesis was the distance between the wrist points and the head, as this would allow the algorithm together with the object in hand signal to determine whether or not a phone is being used for talking.

This signal, HandCloseToHead was created using the viewing direction, origin 3d point and each of the wrist points coordinates.

$$\text{distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{5.3}$$

The variables used in the equation are explained as follows:

x_1, y_1, z_1 The coordinates of the first point in 3D space.

x_2, y_2, z_2 The coordinates of the second point in 3D space.

distance The Euclidean distance between the two points.

5.7 Model development

With the signals in place, it was time to build the models that would recognize the different secondary tasks. The architectures chosen were a simple rule based algorithm and four variations of a random forest model. Every algorithm was developed in Python. For the modelling, 16 of the 34 participants were labelled, and the labelled data was separated in training and evaluation subsets of the data. 12 of the 16 were used for training and 4 of the 16 were saved for evaluation.

5.7.1 A simple rule-based model

The simple rule based algorithm consists of a main class and the three classes ObjectHandMemory, WristHeadDistanceMemory and GazeWristMemory. The main class is iterating over each row in the current secondary task event file and feeds each row to the different memories. Each memory then makes its computations and returns a prediction that is then used by the main class to determine the current task of the driver.

ObjectHandMemory The object hand memory is a class responsible for remembering the position and type of objects in the driver's hands. It is using the object in hand signals objectType, objectPresent and the positions of the wrist. This allows the algorithm to remember the last position of the wrist when an object in hand was identified, which is useful if objects get occluded by head or steering wheel and increase robustness.

Algorithm 2 Class ObjectHandMemory

Initialize with:

- initial_lifetime (default: 600)
- quality_threshold (default: 0.5)
- memory (an empty dictionary)

Function process_row(row):

- Update lifetimes of existing objects in memory
- Check each hand:
  - If hand has a valid object type and quality:
    - Update or add hand object in memory
    - Reset its lifetime to initial_lifetime
- Return predictions from make_prediction()

Function make_prediction():

- Initialize hand_predictions
- Iterate through memory:
  - Update hand_predictions with longest lifetime objects
- Return hand_predictions

Function get_object_type(value):

- Match value:
  - Return corresponding object type

WristHeadDistanceMemory The wrist head distance memory is responsible for keeping a memory of the distance between the wrist points and head origin. While being able to determine where the wrist points were last seen, it also keeps a trailing window of the N last seen positions of the wrists. The window is used to provide a mean of the N last seen distances, which is essentially filtering the signal and makes it less sensitive to a flickering signal.

GazeWristMemory The gaze wrist memory, being the most complex of the memory classes, is responsible for keeping a memory of gaze wrist intersections, utilizing the ray-sphere intersection test described above. It keeps a memory of the N latest intersections and when a phone was last seen in corresponding hand. Similar to how

Algorithm 3 Class WristHeadDistanceMemory

Initialize with:

- distance_threshold (default: 0.8)
- window_size (default: 50)
- quality_threshold (default: 0.5)
- memory (an empty dictionary)

Function process_row(row):

- Determine better quality viewing direction (CMS or DMS):
  - Set viewing_origin based on quality
- For each wrist (9 and 10):
  - If wrist quality < quality_threshold:
    - Continue to the next wrist
  - Else:
    - Calculate viewing-wrist distance
    - Create key as "hand_0" for wrist 9, "hand_1" for wrist 10
    - If key is in memory:
      - Update last seen distance
    - Else:
      - Initialize key in memory with last seen distance and empty mean window
    - Append current distance to mean window
    - If mean window exceeds window size:
      - Remove oldest distance
    - Calculate mean distance
- Return predictions from make_prediction()

Function make_prediction():

- Initialize hand_predictions with {"hand_0": False, "hand_1": False}
- For each key and value in memory:
  - If mean distance < distance_threshold:
    - Set hand prediction to True
- Return hand_predictions

the wrist head distance memory provides a mean distance of the previous window, gaze wrist memory is using a combination of a similar mechanic together with the object in hand to create predictions.

Algorithm 4 Class GazeWristMemory

Initialize with:

- intersection_window_size (default: 120)
- phone_seen_window_size (default: 120)
- quality_threshold (default: 0.5)
- memory (an empty dictionary)

Function process_row(row):

- Update phone seen status for each hand
- Determine better quality viewing direction (CMS or DMS)
- For each wrist:
  - If wrist quality < quality_threshold:
    - Continue to the next wrist
  - Else:
    - Calculate offset wrist origin
    - Perform gaze-wrist intersection test
    - Update memory with intersection result
    - Update intersection memory window
    - Calculate mean distance
- Return predictions from make_prediction()

Function make_prediction():

- Initialize hand_predictions with {"hand_0": False, "hand_1": False}
- For each key and value in memory:
  - Check if phone was seen and mean distance is positive
  - Update hand prediction if both conditions met
- Return hand_predictions

Function get_object_type(value):

- Match value:
  - Return corresponding hand side

5.7.2 Random forests

For comparison, several random forest models were trained using different preprocessing methods. Those methods are rolling mean/median, lagging columns and original data. For each model trained, the amount of rows with event label active was balanced with an equal amount of rows with event inactive. The python library sklearn was used as a framework for the models.

Base model The base model was using the raw signals from the logs, frame by frame, with the event labels as target.

The data used for the base model was the following.

Data	Description
lHand/activity/value	Left hand activity
rHand/activity/value	Right hand activity
lHand/objectPresent/value	Left hand object present
lHand/objectType/value	Left hand object type
rHand/objectPresent/value	Right hand object present
rHand/objectType/value	Right hand object type
lHandOffWheel/value	Left hand off wheel
rHandOffWheel/value	Right hand off wheel
lWrist/value/1	Left wrist height coordinate
lElbow/value/1	Left elbow height coordinate
rWrist/value/1	Right wrist height coordinate
rElbow/value/1	Right elbow height coordinate
attendDist/durationMs	Distraction length
attendDist/value	Distraction value
longDist/durationMs	Distraction length
longDist/value	Distraction value
shortDist/durationMs	Distraction length
shortDist/value	Distraction value
lWristDistHead	Left wrist head distance
lWristIntersect	Left wrist gaze intersection
rWristDistHead	Right wrist head distance
rWristIntersect	Right wrist gaze intersection

The output from trackergui already provides predictions such as object in hand, and hand activity. Those signals are already quite robust, and provide a very good insight into the activity of the driver. The signals are not perfect though, and often lose track of the features that it is tracking. Applying an additional late fusion of the signals may therefore only hope to increase the robustness of the secondary task tracking by a tiny amount.

Rolling mean and median To capture the aspect of time in the data, the raw signals were aggregated over the nth past rows similarly to how the simple rule based model handled time. For every row of data, compute the mean or median for the nth past rows and add it to a new column.

Algorithm 5 Function apply_rolling_filters

Function apply_rolling_filters(dataframe, columns, window_size):

- Copy dataframe to new_dataframe
- For each column in columns:
  - Calculate rolling mean with specified window size
  - Calculate rolling median with specified window size
  - Create new column names for rolling mean and rolling median
  - Add rolling mean and median to new_dataframe
- Drop first window_size rows from new_dataframe
- Return new_dataframe

Lagging columns Another way to add a measurement of time is to add so-called lagging columns. Similar to how rolling filters are applied, lagging columns are columns with data from previous rows. In this implementation each column, for example *lWristDistHead*, is used to create new columns as described in the pseudocode below using 10 steps of 10 rows, which results in 10 new columns of each column, *lWristDistHead_lag_10*, *lWristDistHead_lag_20* and *lWristDistHead_lag_30* and so on. Pseudocode for this can be observed in 6.

Algorithm 6 Function create_lagged_columns

Function create_lagged_columns(df, columns, n, x):

- Copy df to result_df
- Initialize empty list lagged_cols
- For each column in columns:
  - Create lagged columns:
    - Shift column by $i \times x$ for i in range 1 to n
    - Name each lagged column as 'column_lag_$i \times x$'
    - Add these lagged columns to lagged_cols
- Concatenate original df with lagged columns
- Return result_df excluding the first $n \times x$ rows

Choice of events to predict For the models to train, a subset of the labelled events were chosen. The reasoning for this was that some of the events had very little data, and others were not perfectly labelled. Event *OtherBehavior* which refers to an unidentified task, for example scratching ear, was properly labelled during the events when the participant was asked to do the secondary task of scratching ear. This also happened naturally in situations where the participant actually had to scratch their ear, and this was potentially missed in the labelling. Therefore, it was deemed hard for the models to be able to predict this. Other tasks, like *Eating* or *Drinking*, were left out due to lack of data. Compared to *InteractingPhone*, which was present during many of the secondary task events, *Eating* and *Drinking* only occurred twice for each participant. The events that were chosen for training the models were *InteractingPhone*, *PhoneToEar*, *ObjectInHand*, *PhoneUseDynamic*, *PhoneUseStaticHigh*, *PhoneUseStaticLow*.

Feature importance analysis In order to answer RQ1-a, a feature importance analysis was done. The 10 most important features of each model type, i.e. rolling window and so on, were aggregated over every secondary task event in the evaluation subset of the data.

6

Results

6.1 The dataset

The dataset that was recorded and produced for this project contained 34 participants performing 28 different secondary tasks with variations. Demographic data was not collected and the participants were mostly Smarteye employees. The data from each participant contained a CMS video recording and a DMS video recording with associated log files.

6.2 Machine learning models

Several random forest models were trained using different data preprocessing steps as described in the procedure section, and below follows the results necessary to answer the research questions.

6.2.1 Most important features

The results of the feature importance analysis can be seen in figure 6.1. The figures represent examples of the mean importances over the different event classes. It is clear that the height values for the left and right wrist are the most important input features. Other important features are hand activity, wrist distance to head, object type in hand and object present in hand. The figures in 6.1 represents the top 10 important features on different models attempting to predict certain events. The top row in each figure are the most important ones. Those insights may be used to answer RQ1-a, *Which sensory signals can be important in identifying and classifying phone usage in the context of driving?*, as they describe which of the features in the input vector to the models was most decisive in classifying the different types of events.

Figure 6.1: 10 most important input features for *PhoneToEar*, *ObjectInHand*, *PhoneUseDynamic*, *InteractingPhone*

6.2.2 Model comparison figures

To compare performance of the different random forest models trained on different types of events, three recorded secondary tasks events were chosen from the evaluation dataset. The figures in 6.2, 6.3, 6.4 describe the classification accuracy on different types of labelled events, including the model that is not using any form of rolling time window. Those results can therefore be used to answer both RQ2, *How may time be represented in filtering sensory signals?*, and RQ3, *Does the dimension of time improve the performance of models predicting secondary tasks?*, as it compares the performance of different time representations in the data. Please refer to chapter 7 for further discussion on those results. In the figures the x-axis describes the time in frames and the y-axis describes whether or not the model is predicting 0 or 1 for the event. 0 is event not active and 1 is event active. The red field represents when the event is active based on the labelling of the data, and is therefore ground truth.

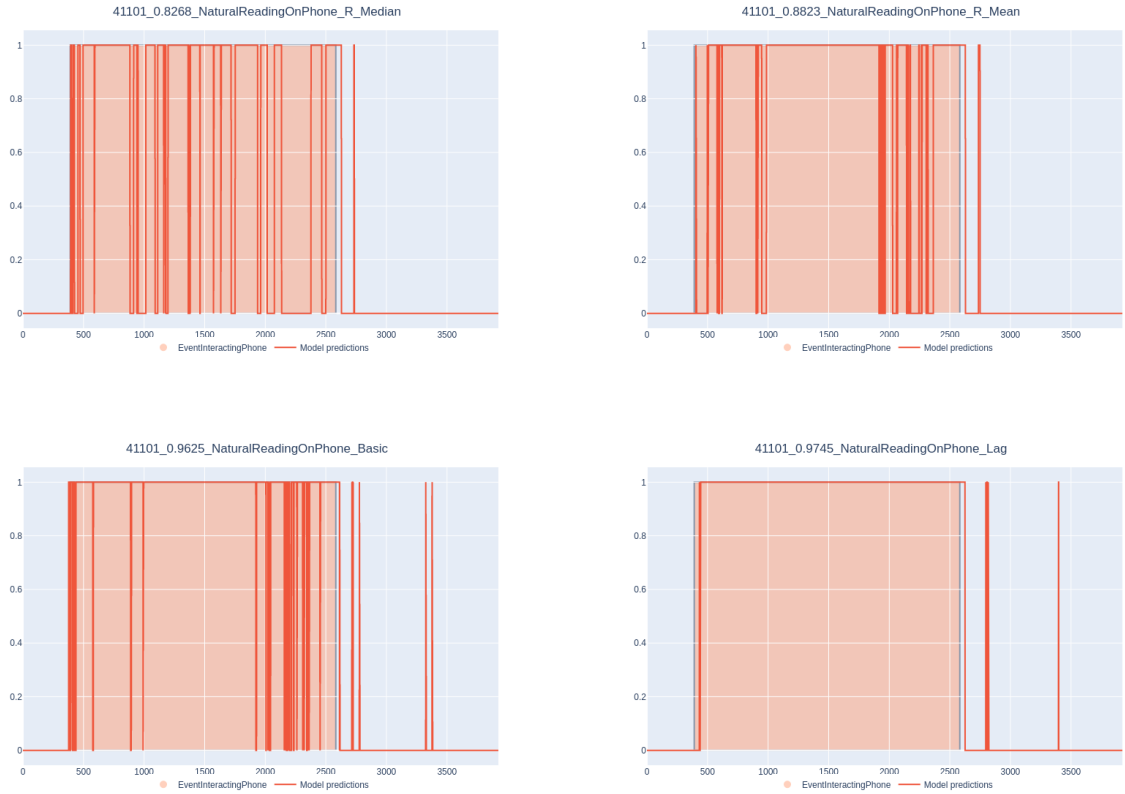

Figure 6.2: 4 models predicting InteractingPhone on task NaturalReadingOnPhone.

InteractingPhone The tasks *NaturalReadingOnPhone* was used for *InteractingPhone* as seen in 6.2. It is clear that all the different random forest models perform well on the task, with the lag model doing slightly better than the basic one. Worth noting is that the best performing models also seem to produce the highest amount of false positives.

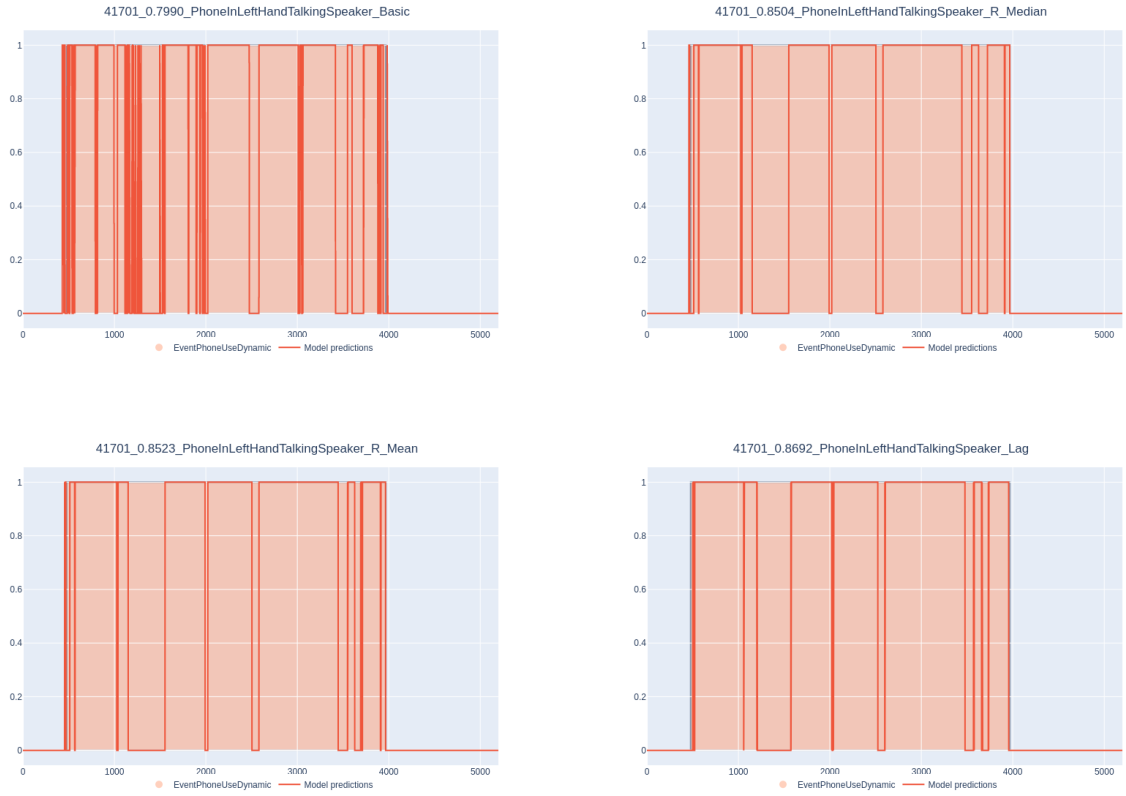

Figure 6.3: 4 models predicting PhoneUseDynamic on task PhoneInLeftHandTalkingSpeaker

PhoneUseDynamic *PhoneInLeftHandTalkingSpeaker* was used for *PhoneUseDynamic* as seen in 6.3. The percentage performance suggests that the basic model performs significantly worse than the different models that have the aspect of time included. Similar to the *InteractingPhone* event, the lag model has the highest accuracy on the task.

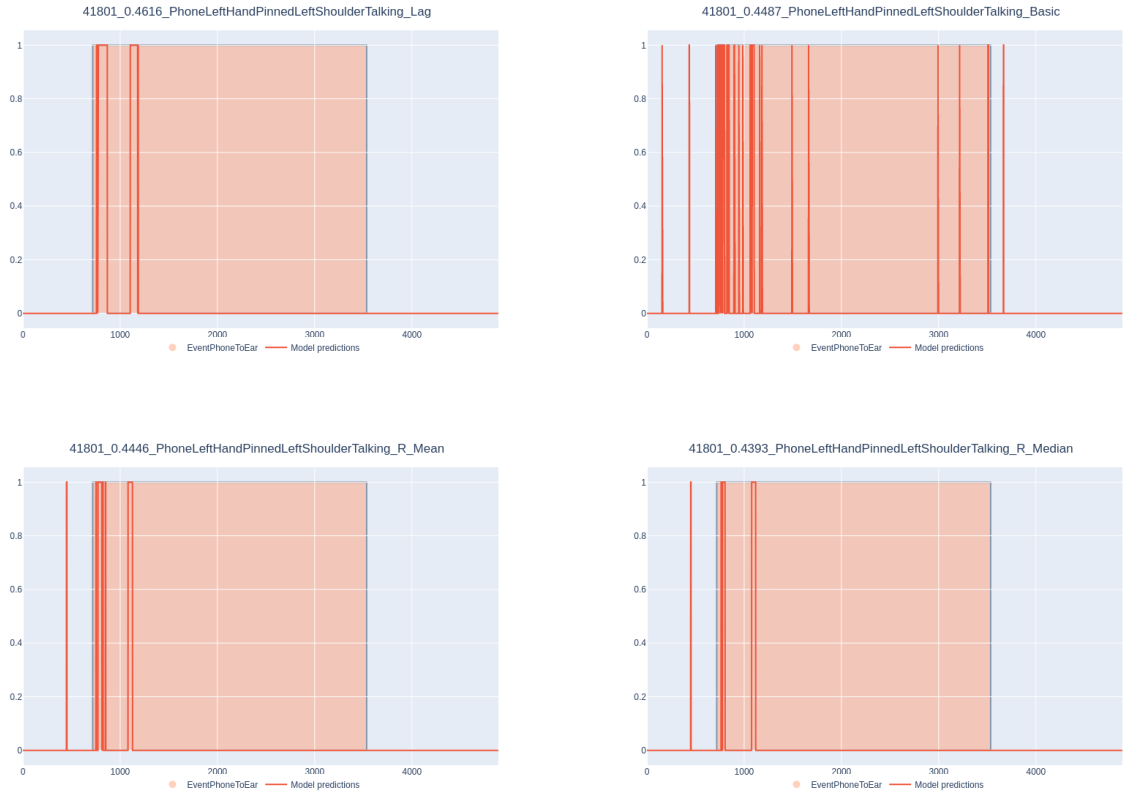

Figure 6.4: 4 models predicting PhoneToEar on task PhoneLeftHandPinnedLeftShoulderTalking

PhoneToEar *PhoneLeftHandPinnedLeftShoulderTalking* was used for *PhoneToEar* as seen in 6.4. Serving as an example of the worst performance over all models and tasks, *PhoneLeftHandPinnedLeftShoulderTalking* proves to be a great challenge for the models. The basic model seems to occasionally make correct predictions later on in the recording, but is also producing more false positives than the other models. It is worth noting that every model seems to perform worse than random, which implies that there might be an issue of over-fitting.

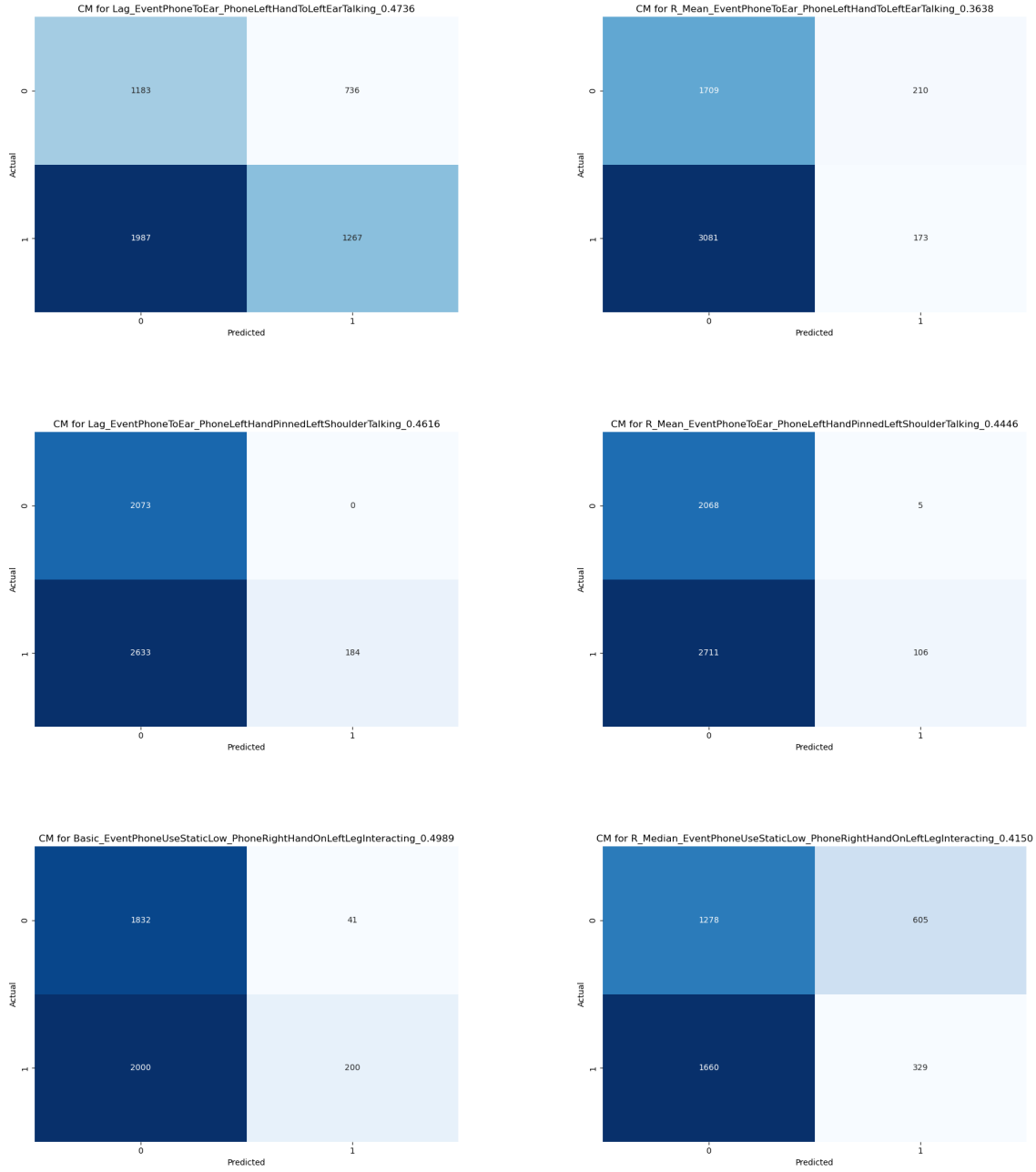

Figure 6.5: Confusion matrices of the most common false negatives

6.2.3 Common false negatives

An excerpt of the secondary tasks most commonly producing false negatives can be seen in figure 6.5. Those are the events *PhoneToEar* on tasks such as *PhoneToEarTalking* and *PhoneLeftHandPinnedLeftShoulderTalking*. The bottom left cell represents the false negatives in the predictions. A description on how to read confusion matrices can be seen in fig 6.1.

Those results can be used to answer RQ1-b, *which phone-related secondary tasks are*

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Positive (FP)
Actual Negative	False Negative (FN)	True Negative (TN)

Table 6.1: 2x2 Confusion Matrix description

most prone to produce false negatives?, as it describes the event classes that are most prone to producing false negatives. Please refer to chapter 7 for further discussion on those results.

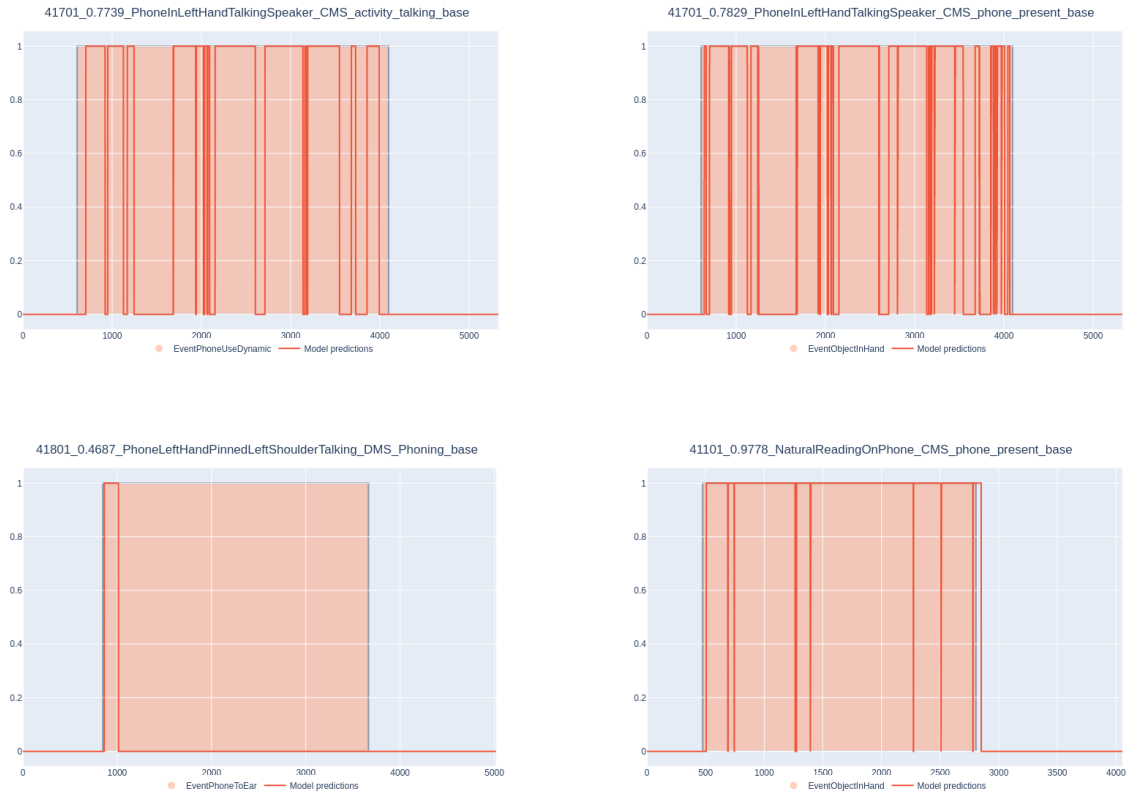

Figure 6.6: Basic model performance excerpt

6.2.4 A simple rule-based algorithm

To compare the simple rule based algorithm with the random forest, some of the better examples can be seen in 6.6. The simple rule base algorithm performs significantly worse than the random forest models on all tasks except *PhoneLeftHandPinnedLeftShoulderTalking* and *NaturalReadingOnPhone* where the accuracy is comparable. In the plots the x-axis describes the time in frames and the y-axis describes whether or not the model is predicting 0 or 1 for the event. 0 is event not active and 1 is event active. The red field represents when the event is active based on the labelling of the data, and is therefore ground truth.

6.3 Characteristics that define phone usage

The characteristics of the final definitions of phone usage that would guide the data collection of the modelling was produced through an iterative process that can be described using the following steps.

Step	**Description**
Initial proposal	An early description of the scope was drafted and presented to the stakeholders at Smarteye
Scope workshop	Based on literature and expert interviews a list of secondary tasks was decided on
Defining secondary tasks	A detailed list of the secondary tasks to include in the data collection was produced
Defining variations of secondary tasks	Potential variations in the secondary tasks was discussed and added to the list
Final definitions	The final list of secondary tasks in 5.2 was presented to stakeholders at Smarteye and accepted

Please refer to chapter 7 below, where extensive discussion is provided on how this relates to answer RQ1, *what characteristics should be considered when defining phone usage in the context of driving from the perspective of driver distraction warning system?*.

7

Discussion

Below follows a discussion on the design of the data collection, the modelling of the data, how human-AI interaction relates to this project and common issues with AI infused systems. Since RQ1 is the main question of the thesis, it will be discussed last.

7.1 Modelling

7.1.1 Most important features

RQ1-a: *Which sensory signals can be important in identifying and classifying phone usage in the context of driving?*

To answer RQ1-a, we may refer to 6.1. First and foremost, it seems to be the height coordinate of the wrist points in particular that allows the model to predict the different secondary task events. While this may seem surprising, it is obvious that the position of the wrist does correlate with the activity. For example, during a *PhoneToEar* event, the hand is holding the phone at a certain height during the event. The model may therefore learn that this is a common feature in the input vectors that should be recognized as *PhoneToEar*. Furthermore, the signals hand activity and object in hand is also important, since those are likely very useful when predicting any task that is including essential components such as holding or not holding an object in hand. Those signals are likely based on object recognition models, and the most important raw sensory signal can therefore be assumed to be a camera feed.

7.1.2 The dimension of time

RQ2: *How may time be represented in filtering sensory signals?*

The dimension of time was incorporated into the modelling in two ways, by rolling windows and lagging windows. Considering the 60 frames per second of the recordings, both the rolling windows and lagging windows covered a relatively short time period. The rolling window, with a window of 30 frames, did only include half a second of frames. This explains the loss of accuracy in figure 6.2, for the R_mean and R_median models. As stated previously, *InteractingPhone* does include stretches of time during which the phone is not interacted with, even though it may still be

in the hand of the driver. A larger window may have resolved this issue to some extent, but it is reasonable to assume that this would have resulted in the model not stopping to predict *InteractingPhone* after the event is actually over, which can be seen in 6.2 as well, if looking closely to where the pink block ends. The lagging column model on the other hand performs significantly better on this task, and there are hardly any false negatives at all during the entire event. This may be explained by the driver likely not focusing on the task of driving for more than 100 frames before returning to the secondary task, and the memory of *InteractingPhone* being included in the 100 frame window. The basic version of the model seems to perform equal to the lagging model even though it does have a problem with many false negatives during the event. While the basic model does not suffer from the penalty of being slow to react due to the lagging window, the fact that it is only able to make its predictions frame by frame does make it less stable to missing and flickering signals in the input vector.

A more general event such as *PhoneUseDynamic* does prove to be a more challenging event to predict for the models, and the accuracy is significantly less for all models in the provided example in 6.3. This is likely due to the great variation in *PhoneUseDynamic*, as it is essentially defined as any moment where the driver has their phone in hand. This may include talking, interacting with and just holding the phone in hand. Referring back to the most important features in 6.1 with this in mind, it is clear that the height of the wrist point will not be able to help in predicting a behavior that varies so much and that the only signal that should be active during such an event is seeing an object in hand.

One of the most challenging events to predict seems to be *PhoneToEar*, as it is often the case that the phone gets occluded causing the algorithm to lose the object in hand signal. This does not seem to be a problem as long as the hand is holding the phone that is close to the ear, as the models may use the wrist height coordinate to infer what the driver is doing. In 6.4 the driver does have their phone to their ear, without supporting it with their hand. This results in an accuracy less than chance, and we can see that the previously best performing model type, utilising the lagging window, has a 46% accuracy. What makes this curious is that if instead of having a trained model the algorithm just outputted 0 or 1 randomly, it should have balanced out on a 50% performance. With this in mind, it is clear that some feature in the input vector is causing systematic false negatives. According to the feature importance analysis in 6.1, the strongest predictors are wrist height, object in hand and hand activity. In the case of the type of task in 6.4, where the driver pins their phone between their ear and their shoulder, and often holds the steering wheel with both hands, it is obvious that the model gets confused with the active signals and therefore makes incorrect predictions. This may be resolved by extending the input vector to the models by adding information of the current head rotation, as this would likely improve the performance in edge cases such as this.

RQ3: *Does the dimension of time improve the performance of models predicting secondary tasks?*

Overall, the lagging window models seem to perform the best. In the examples listed

in the results section it does yield a higher prediction accuracy than the models that are not incorporating time. This is likely because of two reasons. First of all, its window is covering a larger time period. As mentioned previously, the rolling window models are only covering 30 frames, while the lagging window models are covering 100 frames. It is likely that loss of signals due to occlusion and similar issues are extending beyond 30 frames and times, which may cause a series of incorrect predictions. Furthermore, the lagging window models can be expected to be less sensitive to noise in some scenarios, since it only samples one frame for every ten frames.

The solutions of this project does incorporate time to some extent, but in the most basic way possible. It is reasonable to assume that a dynamic approach such as a LSTM solution would be able to perform better on the data available, provided that more data was available to train such a network. The reasoning behind this is that LSTM networks are able to weigh its memories, giving more recent memories more importance and therefore more say in the final predictions.

7.1.3 Common false negatives

RQ1-b: *Which phone-related secondary tasks are most prone to produce false negatives?*

To answer RQ1-b, an excerpt of the worst performances of the different models is presented in figure 6.5. The first row of confusion matrices describes the predictions during a secondary task event where the models are attempting to predict *PhoneToEar*. The most common prediction is that there is no phone to ear, even though a major part of the frames is composed of the driver having the phone to their ear and talking. This is likely caused by a loss of signal due to an occluded phone, which based on the feature importance analysis is one of the most important features in the input vector. The next row contains two more events including *PhoneToEar*, during a task with the phone pinned between ear and shoulder. The explanation of the low accuracy here is like the same as mentioned earlier, discussing the figures in 6.4, which is that the main important features point towards not having a phone to ear, and that this is causing an accuracy lower than chance. Similarly, on the last row of the confusion matrices, the models are attempting to predict *PhoneUseStaticLow* which is an under-represented event in the data. Referring to the labelling rules in 5.4 *PhoneUseStaticLow* is described as the phone reaching a low static position, which in practice is that the driver is placing their phone on their leg or in their lap and leaving it there while talking using speaker phone or interacting with it. Similarly to how the models lack information as to predict the events where the phone is pinned between the ear and the head of the driver, this event lacks essential information on where the phone is currently located. The model can not use the information on if an object is in the hand of the driver, since it lays still in a static position, and it does not have the means to localise the phone. Adding this information to the input vector would likely ameliorate the issue.

To conclude, the models are only as insightful as the data allows them to be. There is clearly some missing information in the input vectors to the models, which ends up

causing many false negatives for some of the event classes that the models attempt to predict.

7.2 Defining phone usage

RQ1: *What characteristics should be considered when defining phone usage in the context of driving from the perspective of driver distraction warning system?*

The definition of phone usage in this project was derived from literature and how phone usage is tracked by the available systems. This was extended with the findings and discussions in workshops and the final definitions for this particular project can be seen in 5.2. In order to answer the main research question, we may summarise the findings in the result section relating to the sub questions. The most important features in identifying and classifying phone usage in the secondary tasks chosen for this project can be seen in figure 6.1, and the results point towards hand position, the hands distance to head and whether or not the driver is holding an object in their hand. Furthermore, the most common false negatives are the events where the most important features are lost due to either occlusion or similar reasons for lost feature signals. Putting all the pieces together, it is reasonable to say that the characteristics that should be considered when defining phone usage are those that are describing the posture of the driver, i.e. hand position and hands distance to head, and the content of the hands. While this comes a long way towards defining the characteristics of a well performing model, it must be extended with the fact that those characteristics have their strengths and weaknesses. To answer RQ1, the characteristics that should be considered when defining phone usage in the context of driving from the perspective of driver distraction warning system is those that can be tracked by the available systems, in this case hand position, hands distance to head and the content of the hands, but also unknown features that have not been identified in this project as this may be assumed to provide a stronger foundation for the models to use when classifying different secondary tasks, to allow for a loss of some of the signals from time to time.

7.3 A "wicked" problem

There is no definitive formulation of a "wicked" problem. A problem can be described in a complete way that encompasses the entire problem space and provides the receiver with sufficient information to solve it. This is not possible for a "wicked" problem, due to it residing in an unknown domain, or an ever-changing one. In the context of this thesis, it is clear that driver monitoring does not have a clear definition in the sense that neither legislation nor recommendations based on science and statistics are able to formulate a finite problem space. It is partly understood and accepted, the driver must be monitored until full automation is reached, but how to do this and what expressions and features to monitor is not close to being agreed upon.

Referring back to 3.1 it is clear that many of the challenges in this project have

had characteristics of "wicked" problems. The definitions necessary to carry out the practical work have been far from exhaustive, and did only cover a small portion of what phone usage in the context of driving really is. The problem space is vast and the limitations have been many. Despite being constrained by time and amount of data, it is not clear that an infinite amount of time and data would have reduced the "wickedness" of the problem at hand, due to challenges in defining human behaviour. To some extent, one major contributing factor to a "wicked" problem in design is the fact that it is nearly impossible to pinpoint universal traits in humanity, and that it therefore is very hard to produce accurate definitions of human behaviour. Assuming that this would be possible, or if a designer would settle for a "tamer" version of the problem which would be to only design for a small sample of humanity, other problems would arise. The next issue would be that "wicked" problems such as modelling human behaviour does not seem to have a natural stopping rule. As mentioned in 3.1, it may be possible to produce a model that is able to predict the drivers behaviour with a 100% accuracy. The key is that the solution to a "wicked" problem requires that the problem is "tamed" by constraining it to a controlled portion of the problem space. Using the examples of modelling, the algorithm can only learn what the developer tells it to, and will forever be constrained to the definitions that have guided the data collection and the labelling of the data. Furthermore, there are an endless amount of potential solutions to the problems dealt with in this thesis, which also makes it a "wicked" problem. From the choice of models to the setup of the data collection and to the analysis of the data. The problem is unique, in the sense that every group of researchers approach it from their own angle. While legislation may set relatively clear requirements, the way to meet them is infinite.

7.4 Human AI-interaction

The practical work that this project has been composed of may be observed as a case-study in designing for Human-Centred AI (HCAI). The basis for the decisions made throughout is based on the guidelines presented by Amershi et al. [38] in 2019. It must be said that what this project has produced is not a complete interactive system, but rather only the sensing side of one. Hornbæk and Oulasvirta provide a suitable definition of what interaction is, and calls it *"interaction concerns two entities that determine each others behaviour over time"* [48]. Following a Research for Design approach (section 4.1), this project has only delivered one side of the interaction circuit. Therefore, to fully understand how a deployed driver monitoring system behaves, and to guide design choices made through-out the project an imagined complete interaction system has been designed for. This has been important, as it has forced a certain level of clarity to the more technical parts.

7.4.1 An imaginary AI-infused advanced driver distraction warning system

As mentioned earlier in this report a driver monitoring system is there to provide the driving system, the car, an understanding of the drivers state and assist the driver in behaving safely and in line with legislation. One example of this may be to detect drowsiness and in some way make the driver take a break from driving. In older systems this was decided by a timer, sending out a warning after a predefined period of time had passed. While this likely provided some measurement of increased safety, it was a simple and non-interactive system. With the advent of deep learning and machine learning, building systems that would allow the driving system to better understand the driver became possible. It is clear that this is not a trivial task, but systems are constantly improving together with algorithmic advances and new findings in the field of AI.

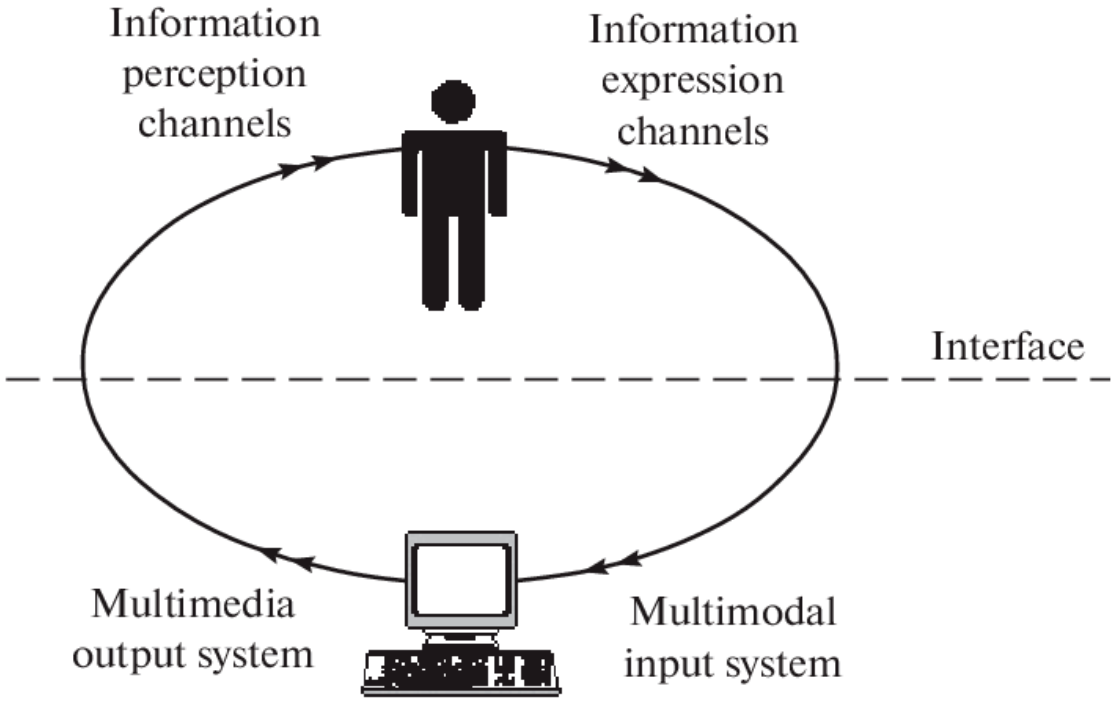

Figure 7.1: A diagram of multimodal human-computer interaction as depicted by Karpov and Yusupov [49].

This is the sensing part of Karpov and Yusupov's interaction circuit. The driver is expressing information which is observed and processed by the driving system, and the system then behaves and reacts according to the sensed state of the driver to moderate their behaviour. This can be done in many ways, from carefully nudging the driver with beeps and lights to presenting a visual warning in the interface. As mentioned in the introduction, this is a high-risk environment where the system may easily distract the driver further and therefore put both the driver and other people at risk of harm. Another potentially less distracting approach to this is to use natural language as the interface for such interaction. This may either be done

with recorded sound bites such as "*please take a break soon*" or "*you are currently speeding*". While this may be considered a valid interactive interface, there are several problems with such a solution. Over time, the driver may experience a lack of personalised interaction and that the system is limited on a technical level. To get around this, one may instead imagine the system's output being expressed through a large language model, an LLM. This kind of technology comes with its own set of limitations and problems, but for the sake of the argument one may make some assumptions. In this imagined system, the AI infused system is sensing the state of the driver by using CMS and DMS. The stream of information is being processed and the system is providing a clear and detailed prompt to the LLM which is fine-tuned to be able to behave as an advanced driver warning system. This would presumably be experienced as more personalised and dynamic by the driver.

The warnings provided by the system may look something like below:

Prompt	**Output**
The driver is currently focusing on the task of driving, but has been driving for a long time and is drowsy.	*You've been driving for a while now. How about a quick break to stay sharp and alert?*
The driver is talking on their phone, while also focusing on the road ahead.	*Remember to keep youtemporalr full attention on the road. Can you finish your call later?*
The driver is reading something on their phone, repeatedly looking away from the road.	*Eyes on the road, please. Your safety is the prioritylet's save the reading for later.*

7.4.2 A hypothetical Human-Centred AI driver-monitoring system

Using the hypothetical driver-monitoring system below, it is now possible to exemplify the Human-AI Design guidelines by Amershi, Weld, Vorvoreanu, *et al.* [38]. The aim of this imaginary setup is to contextualise the design choices that went into the design and development of the algorithmic models, and to demonstrate what the technology can enable in terms of new driving experiences. As mentioned in the theory section on Human-Centred AI 3.2, the 18 guidelines are separated into four parts. The guidelines are broad and high-level, which makes them applicable to AI-infused design in general, while also allowing for some interpretation. Many of the guidelines relate to the imagined part of the proposed system described above, but will be included as a basis for discussion. The guidelines may also be rephrased to fit the context of driving.

Initially The first section of the guidelines relates to the initial interaction between the driver and the user. The system must make it clear to the driver what it can do, and to what extent it may be expected to be successful in the task. The interface between the driver and the system is in this example natural language, and the driver must in some way be made aware of the sensing that is going on, through cameras and such, and that this observation will allow the system to understand their current state. While this part is not included in the work of this project, it

does provide a high-level description of what the systems understanding may be based on which would be possible to share with the driver. The end signals, that would be used to prompt the LLM, are described in such a way that it is clear what they refer to. *Interacting*, for example, tells whether or not the driver is currently interacting with their phone and it is possible to describe what features this signal is mainly composed of by the feature importance calculations. There is of course a balance to this. If the driver perfectly understands the system that is put in their car due to legislation against their will, this may teach the driver how to trick the system in order to not be disturbed by it. This is likely not a problem, since the system must have a shutdown function, as mentioned in the introduction (Chapter 1).

During interaction To guide the behaviour of the active online system, there are according to the guidelines two main categories. The system must be aware of the current context and time frame in which the driver acts, and the output provided by the system must be sensitive to the users social and cultural context while at the same time not acting in a way that may be based on stereotypes and biases.

The first part relates to the temporal aspect of the modelling in this project, but would have to be extended in order to fulfil the description of this guideline. As mentioned in the procedure different secondary tasks can only be described over time. The secondary task of *interacting* for example, looked at through the eyes of the system, consists of a series of frames where the driver is either actively interacting with their phone by looking at it and touching the display, but also when pausing the interaction to focus on the task of driving before returning to the interaction with the phone. The system must therefore be able to understand that a secondary task is still active even though nothing that is currently being observed by the sensors is telling it so. At this level, the models developed in this project do take this into account, using a rudimentary memory mechanism that to some extent provides the system with the necessary temporal insights to meet parts of this guideline. What is missing is a broader understanding of both the environment outside of the car, and a deeper understanding of the incentives behind the secondary tasks that the driver is performing. Using the example of *interacting*, it is illegal for the driver to use their phone for texting while driving, but it is allowed to use a GPS for navigation. This may look very similar from the systems perspective, while only the former part should invoke a warning.

The second part of the *during interaction* guidelines concerns the output of the system, and its sensitivity to the driver's person and context. At first glance, this may sound like it refers to the imaginary part of the proposed system, for example providing the LLM with guard rails that would make sure not to disregard the drivers social and cultural context. This would likely be part of the solution, and can be considered a filtering mechanism of the outputs from a LLM as a band-aid for the chaotic nature of models trained on immense amounts of data. This will be further discussed below . This could of course also be handled in the sensing of the system, and the features extracted from the raw sensory data. The question is then whether it is ethical to build and deploy a system that has the means to recognize and identify features like age, gender and race, in order to be able to incorporate a

more "personalised" interaction. Assuming that such modelling would be possible on a technical level, how may such a dataset needed for training the models be acquired in an ethical way? One could argue that this guideline is not applicable in all kinds of HAI, and that it would be preferable to try to identify a more universal mode of interaction, if that is possible. This would entail exploring more general ways for the LLM to express itself, that would be well received by drivers of different cultures and backgrounds.

When wrong The *when wrong* guidelines involve the user's control over the system, and the ability to understand the system's behaviour and to correct it if necessary. While this may not be obvious in the context of this project, it does illuminate an important aspect of the interaction that is being designed for. One may separate a human being's expressions into conscious and subconscious output signals. If the interaction is taking place on a computer, and the interface is the screen and the keyboard, the user is in control of their input to the system and it is therefore possible for the system to express the reasoning and incentive behind its behaviour by referring to the user's input. In this project, the sensory signals that are going into the system are not necessarily in the control of the driver. The system is interpreting the drivers behaviour and reacting as it is designed to do. There are therefore two obvious solutions to this, either the driver has the ability to quickly turn the system off when it misbehaves, or include additional information in the natural language warnings that the system outputs. Referring back to how the the high-level insights that the system developed in this project is phrased, like for example *object in hand* and *interacting*, it would be possible to design the imagined part of the proposed system in such a way that it explains why it for example asks the driver to take a break or put away their phone. This would of course also be a balancing issue, as to not overload the driver with too much information. One solution would be to use voice as a mode for input to the system. This would allow the driver to ask the system to elaborate on warnings.

Over time The last set of guidelines relates to how the system should develop and adapt over time, to better assist a recurring user. This can be understood on several levels. Firstly, the system may keep a short term memory of the current session, in the way the system internalises secondary tasks such as *interacting*. Secondly, the system could possibly keep a long term memory of the drivers behaviour and also remember the specific driver by face recognition. This would allow it to remember preferences such as how its warnings may affect the driver, and to understand the drivers baseline. Lastly, this could be incorporated into the model itself, by retraining it on collected data. Each level is problematic in different ways, which will be further discussed in the ethics section.

7.4.3 Ethical discussion

The ethical considerations of this project are discussed below.

Biases One of the most common critiques against AI infused systems in general is the biases contained in the data that has trained the models [50]. There exists misconceptions that a diverse dataset, i.e. including a balanced amount of people

of different races, genders and age groups, will solve the bias issue. While this is important to have in mind, it does not result in unbiased models. The lies in the collection of the data, and in the case of this project, how the data collection was designed. The choice of secondary tasks, and the definition of those, was done without diversity and variation in the potential drivers ability in mind. This project did lean heavily on convenience recruitment for the data collection, as it was a proof of concept to eventually expand on later if deemed successful. Therefore, a diverse data collection was not ensured. This does illuminate a broader issue in ethical design in general, which is the amount of time and effort is needed to truly design for humanity as a whole. Taking a master thesis as an example, which does suffer greatly from time constraints, it is challenging to fit ethics into it, without it ending up being just a footnote. Granted, ethical considerations is an essential part of the final report, but it is not made necessary to keep as a theme through-out the project. To conclude, if the goal is to design and develop systems for humanity as a whole, a large amount of data is needed and great care must be taken when designing the data collection.

Human behaviour Considering what was mentioned in the Human-AI interaction discussion above, it is reasonable to assume that the user will adapt to the system that they interact with. In the case of this project, it would hopefully result in the distracted driver becoming a more safe driver by heeding the warnings expressed by the system. The result might also be that the driver learns how to trick the system. The question is whether or not it is ethical to force human beings into changing behaviour. In this case, it is guided by legislation. As mentioned in the introduction, the EU has decided that every new car constructed after 2026 must have a driver warning system built in. While those laws may be well defined and the requirements clear, there will likely be some room for interpretation for the car manufactures. The enforcement of those laws will eventually be put in the hands of the machine learning algorithms that observe and regulate the drivers behaviour.

Societal impact The assumed societal impact of systems that are developed to increase safety by enforcing behaviour is that it will increase safety. In the case of driver monitoring systems, it may be a step closer to zero deaths in traffic (for those who can afford it) [8]. But as with technology in general, it is a step away from privacy.

Privacy There seems to be a trend in the decline of privacy the more technology advances. Machine learning and AI is integrated into more and more of the technology used daily. Privacy is one of the main concerns, and it relates to the value of the behavioural data generated by people who are using services such as social media and other IT services. The system developed in this project is a closed circuit, and does not record data. The question is then, may it still be considered a breach of privacy if the models only observe? If the system reacts to what it observes, it becomes something more than mere observation. Furthermore, it is reasonable to assume that a complex system such as a driver monitoring system will have to keep some form of a memory of recurring drivers in order to better serve and assist them. Referring back to Amershi, Weld, Vorvoreanu, *et al.*'s design guidelines, this must be clearly communicated with the driver.

8

Conclusion

This project has attempted to identify which characteristics should be considered when defining phone usage in the context of driving. The definitions have been described in terms applicable to an advanced driver distraction warning system and may therefore be considered a contribution to the field of driver monitoring. Based on the requirements from the definitions a simulator setup was designed and used for data collection that yielded a dataset containing 34 participants performing a series of secondary tasks. The data contained video recordings and signal logs, which was produced by proprietary tracking software developed by Smarteye. The data was then analysed, preprocessed and used to train several random forest models. Insights generated by the data included a ranking of the most important signals for recognizing and identifying the previously defined secondary tasks and which secondary tasks are most prone to producing false negatives. Furthermore, different ways to represent the dimension of time in tabular data was explored and the approaches were compared using an accuracy score. To conclude, evidence presented in this report suggests that the most important signals in identifying secondary tasks are the position of the driver's hand, the content of their hands and the activity of their hands. The secondary tasks that are most prone to produce false negatives are edge case variations of common secondary phoning related tasks that do not incorporate holding the phone in hand while using it, such as pinning it between head and shoulder, or placing it on the lap. The framework used for structuring the project was the double diamond in combination with HumanCentred AI.

8.1 Knowledge contribution

The knowledge contribution of this thesis is three-fold. Firstly, the characteristics that should be considered when defining phone usage in the context of driving from the perspective of driver distraction warning system are mainly characteristics that relate to the pose of the driver's body and the content of their hands. Secondly, time may be represented in filtering sensory signals by using rolling windows and lagging columns, and the dimension of time does improve the performance of models predicting secondary tasks. Thirdly, the phone-related secondary tasks that are most prone to produce false negatives are those where the input signals tend to get lost, or where the most important features are not involved in the secondary tasks.

8.2 Future work

The future work of this thesis can be described under the topics data, models, definition and further ethical considerations.

The data As mentioned in both the ethics section and the limitations, the dataset that this project has collected is not controlled for diversity in any way due to convenience sampling. Whether or not this may have affected the performance of the models is not possible to say, but it is reasonable to assume that if a system like this would have been deployed in the real world, it would likely have performed worse for groups that were not included in the dataset and therefore not being able to assist the people in those groups in driving safely. Future work would include collecting a more diverse dataset, and making sure that there is a balance over all groups of humans that are driving cars.

The models There is much work to be done related to the modelling. Different types of model architectures may be tested, for example XGBoost and SVM, which could possibly perform better than the random forest used. Furthermore, more signals should be included in the modelling, to build a better foundation for the models to base their predictions on when some of the signals are lost. This would take some exploration, creating models from signals not used in this project.

The definitions The definitions of the secondary tasks were based on the literature, and the experience of experts at Smarteye. This shares the same issues as the data, regarding diversity. The definitions that were used for the data collection and modelling are likely not covering the entire space of variations that is for example *interacting with phone* if looking to humanity as a whole. One may assume that people have many different ways to conduct the secondary tasks chosen for the data collection and future work should consider different ways in which the secondary tasks differ over different social and cultural contexts. To realize this, a qualitative study may be conducted using participants from different backgrounds, in combination with a scoping review on the subject.

Ethics and societal impact The ethical considerations in this report were only briefly touched upon and should be explored more deeply from the perspective of statutory monitoring and the issues related to the impact on human behavior in the context of driver monitoring systems.

Bibliography

[1] 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," *New England journal of medicine*, vol. 370, no. 1, pp. 54–59, 2014.

[2] T. A. Dingus, F. Guo, S. Lee, *et al.*, "Driver crash risk factors and prevalence evaluation using naturalistic driving data," *Proceedings of the National Academy of Sciences*, vol. 113, no. 10, pp. 2636–2641, 2016.

[3] E. Commisson, *Technical support to assess the upgrades necessary to the advanced driver distraction warning systems*, Available at `https://trl.co.uk/uploads/trl/documents/XPR118---Technical-support-to-assess-the-upgrades-necessary-to-the-advanced-driver-distraction-warning-systems-(6).pdf` (2024/01/30).

[4] E. NCAP, *European new car assessment programme (euro ncap)*, Available at `https://cdn.euroncap.com/media/77138/euro-ncap-assessment-protocol-sa-safe-driving-v1012.pdf` (2024/01/30).

[5] E. Tivesten and M. Dozza, "Driving context influences drivers' decision to engage in visual–manual phone tasks: Evidence from a naturalistic driving study," *Journal of safety research*, vol. 53, pp. 87–96, 2015.

[6] N. Euro, "Euro ncap vision 2030–a safer future for mobility," *European New Car Assessment Programme (Euro NCAP)*, 2022.

[7] I. research. "Regulations - drivers for mandating driver monitoring systems." (2023), [Online]. Available: `https://www.idtechex.com/en/research-article/regulations-drivers-for-mandating-driver-monitoring-systems/30322` (visited on 12/12/2023).

[8] I. P. O. V. ZERO, "Euro ncap 2025 roadmap," 2017.

[9] E. Commisson. "Road safety advanced driver distraction warning systems." (2023), [Online]. Available: `https://ec.europa.eu/info/law/better-regulation/have-your-say/initiatives/13740-Road-safety-advanced-driver-distraction-warning-systems_en` (visited on 07/13/2023).

[10] M. Shahverdy, M. Fathy, R. Berangi, and M. Sabokrou, "Driver behavior detection and classification using deep convolutional neural networks," *Expert Systems with Applications*, vol. 149, p. 113 240, 2020.

[11] A. Guettas, S. Ayad, and O. Kazar, "Driver state monitoring system: A review," in *Proceedings of the 4th International Conference on Big Data and Internet of Things*, 2019, pp. 1–7.

[12] L. Fridman, D. E. Brown, M. Glazer, *et al.*, "Mit advanced vehicle technology study: Large-scale naturalistic driving study of driver behavior and interaction with automation," *IEEE Access*, vol. 7, pp. 102 021–102 038, 2019.

[13] J. Feng and B. Donmez, "Design of effective feedback: Understanding driver, feedback, and their interaction," in *Driving Assessment Conference*, University of Iowa, vol. 7, 2013.

[14] F. Vicente, Z. Huang, X. Xiong, F. De la Torre, W. Zhang, and D. Levi, "Driver gaze tracking and eyes off the road detection system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2014–2027, 2015.

[15] L. Alam, M. M. Hoque, M. A. A. Dewan, N. Siddique, I. Rano, and I. H. Sarker, "Active vision-based attention monitoring system for non-distracted driving," *IEEE Access*, vol. 9, pp. 28 540–28 557, 2021.

[16] M. Jeong and B. C. Ko, "Drivers facial expression recognition in real-time for safe driving," *Sensors*, vol. 18, no. 12, p. 4270, 2018.

[17] L. Fridman, H. Toyoda, S. Seaman, *et al.*, "What can be predicted from six seconds of driver glances?" In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017, pp. 2805–2813.

[18] P. Manjula, S. Adarsh, and K. Ramachandran, "Driver inattention monitoring system based on the orientation of the face using convolutional neural network," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2020, pp. 1–7.

[19] B. Ionescu, V. Suse, C. Gadea, *et al.*, "Using a nir camera for car gesture control," *IEEE Latin America Transactions*, vol. 12, no. 3, pp. 520–523, 2014.

[20] Q. Xiong, J. Lin, W. Yue, S. Liu, Y. Liu, and C. Ding, "A deep learning approach to driver distraction detection of using mobile phone," in *2019 IEEE Vehicle Power and Propulsion Conference (VPPC)*, IEEE, 2019, pp. 1–5.

[21] M. Martin, S. Stuehmer, M. Voit, and R. Stiefelhagen, "Real time driver body pose estimation for novel assistance systems," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2017, pp. 1–7.

[22] M. Martin, J. Popp, M. Anneken, M. Voit, and R. Stiefelhagen, "Body pose and context information for driver secondary task detection," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 2015–2021.

[23] S. Masood, A. Rai, A. Aggarwal, M. N. Doja, and M. Ahmad, "Detecting distraction of drivers using convolutional neural network," *Pattern Recognition Letters*, vol. 139, pp. 79–85, 2020.

[24] M. Çetinkaya and T. Acarman, "Driver activity recognition using deep learning and human pose estimation," in *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, IEEE, 2021, pp. 1–5.

[25] Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis, and F.-Y. Wang, "Driver activity recognition for intelligent vehicles: A deep learning approach," *IEEE transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5379–5390, 2019.

[26] L. Li *et al.*, "Time-of-flight cameraan introduction," *Technical white paper*, no. SLOA190B, 2014.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] T. Akilan, Q. J. Wu, A. Safaei, and W. Jiang, "A late fusion approach for harnessing multi-cnn model high-level features," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2017, pp. 566–571.

[30] P. Khaire, J. Imran, and P. Kumar, "Human activity recognition by fusion of rgb, depth, and skeletal data," in *Proceedings of 2nd International Conference on Computer Vision & Image Processing: CVIP 2017, Volume 1*, Springer, 2018, pp. 409–421.

[31] J. Imran and P. Kumar, "Human action recognition using rgb-d sensor and deep convolutional neural networks," in *2016 international conference on advances in computing, communications and informatics (ICACCI)*, IEEE, 2016, pp. 144–148.

[32] S. Ardianto and H.-M. Hang, "Multi-view and multi-modal action recognition with learned fusion," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, IEEE, 2018, pp. 1601–1604.

[33] E. Ohn-Bar, S. Martin, A. Tawari, and M. M. Trivedi, "Head, eye, and hand patterns for driver activity recognition," in *2014 22nd international conference on pattern recognition*, IEEE, 2014, pp. 660–665.

[34] M. Wu, X. Zhang, L. Shen, and H. Yu, "Pose-aware multi-feature fusion network for driver distraction recognition," in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021, pp. 1228–1235.

[35] N. Kose, O. Kopuklu, A. Unnervik, and G. Rigoll, "Real-time driver state monitoring using a cnn based spatio-temporal approach," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 3236–3242.

[36] M. Hultman and J. Anshelm, "Masculinities of global climate change: Exploring ecomodern, industrial and ecological masculinity," in *Climate change and gender in rich countries*, Routledge, 2017, pp. 19–34.

[37] B. Shneiderman, *Human-centered AI*. Oxford University Press, 2022.

[38] S. Amershi, D. Weld, M. Vorvoreanu, *et al.*, "Guidelines for human-ai interaction," in *Proceedings of the 2019 chi conference on human factors in computing systems*, 2019, pp. 1–13.

[39] C. Frayling, "Research in art and design (royal college of art research papers, vol 1, no 1, 1993/4)," 1994.

[40] W. Gaver, "What should we expect from research through design?" In *Proceedings of the SIGCHI conference on human factors in computing systems*, 2012, pp. 937–946.

[41] J. Zimmerman, J. Forlizzi, and S. Evenson, "Research through design as a method for interaction design research in hci," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007, pp. 493–502.

[42] T. D. Council. "Framework for innovation." (2024), [Online]. Available: http://https://www.designcouncil.org.uk/our-resources/framework-for-innovation/ (visited on 01/30/2024).
[43] T. M. Mitchell, *Machine learning*, 1997.
[44] J. Alzubi, A. Nayyar, and A. Kumar, "Machine learning from theory to algorithms: An overview," in *Journal of physics: conference series*, IOP Publishing, vol. 1142, 2018, p. 012 012.
[45] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar, "Cats and dogs," in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 3498–3505.
[46] T. J. OShea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.
[47] P. R. Bakhit, B. Guo, and S. Ishak, "Crash and near-crash risk assessment of distracted driving and engagement in secondary tasks: A naturalistic driving study," *Transportation research record*, vol. 2672, no. 38, pp. 245–254, 2018.
[48] K. Hornbæk and A. Oulasvirta, "What is interaction?" In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17, Denver, Colorado, USA: Association for Computing Machinery, 2017, pp. 5040–5052, ISBN: 9781450346559. DOI: 10.1145/3025453.3025765. [Online]. Available: https://doi.org/10.1145/3025453.3025765.
[49] A. Karpov and R. Yusupov, "Multimodal interfaces of human–computer interaction," *Herald of the Russian Academy of Sciences*, vol. 88, pp. 67–74, 2018.
[50] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM computing surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

9

Appendix 1