Towards a Real-time System based on Regression Model to Evaluate Driver's Attention

Thiago K. Lago[1], Ernesto Rodríguez González[1], and Miguel Elias M. Campista[1]
[1]*GTA-PEE/COPPE-DEL/Poli – Universidade Federal do Rio de Janeiro (UFRJ)* – Rio de Janeiro, Brazil

thiagokoster@poli.ufrj.br, {ernesto, miguel}@gta.ufrj.br

***Abstract*—The use of mobile devices while driving is one of the major causes of accidents. Thus, this paper aims to create a real-time system to alert drivers of inattention moments, and thus reduce the number of traffic accidents. For this, we use computer vision algorithms to determine the driver's gaze direction which, together with vehicle data such as speed and acceleration, infer whether the driver is distracted. The key idea is to calibrate the drive's head rotation thresholds as a function of vehicle speed and acceleration. If the vehicle is at low speed or low acceleration, the head rotation can be more pronounced. In addition to head rotation, we also use blind eye detection as a criterion for determining distraction. The system uses a temporal sliding window procedure to prevent oscillations between inattention and attention states. The implementation is based on hardware and software architectures composed of cost-effective and open source libraries. Experimental results in controlled and real environments show accurate results and quick detection time.**

***Index Terms*—Computer vision, Safe driving, IoT.**

I. Introduction

The popularization of smartphones and the consequent increase in the number of available applications bring a permanent source of distraction for users, including those driving vehicles. For drivers, the use of smartphones can drastically impact the response time to unexpected events and, consequently, the chances of an accident [1]. In 2019, 8.7% of all people killed in motor vehicle traffic crashes were caused by driver distraction in the United States. This accounts for 3,142 deaths according to the last report released by NHTSA (National Highway Traffic Safety Administration) [2]. These alarming numbers show the importance of safeguarding drivers against distractions.

An alternative to inhibit the use of smartphones while driving is the development of systems able to detect drivers' distractions while in traffic. To handle that, image processing algorithms are typically deployed along with data obtained from the vehicle and the surrounding environment. The system must provide a reliable and fast decision, which depends on many factors, such as the camera, the available sensors, and the computer on which the assessment is carried out.

This paper proposes a system to alert driver's distractions by detecting the head orientation along with data concerning the vehicle acceleration and speed. We, however, do not consider that the driver is distracted when looking down if the car is stopped or when quickly taking a look at the side mirror. To accomplish that, we compare the driver's head rotation angle according to yaw and pitch axes with predetermined maximum thresholds, computed considering regressive models based also on vehicle speed and acceleration. Similarly, we also take into account blind eye detection to infer driver's distraction. To avoid fast changes between distraction and attentive status and false alarms, we use counters to only infer distraction after a certain number of consecutive frames using a sliding window approach. We build a prototype composed of a laptop containing an external camera and a Raspberry Pi to collect the additional data. The system software is implemented in Python using a modular approach, which uses open-source libraries and has many configuration possibilities for finer-grained customization. The proposed system also logs all the data collected in a file for offline analysis and audition. We conduct experiments in two scenarios: a controlled and real one. The obtained results show that the system operates as expected, quickly and consistently inferring distractions even if part of the input parameters are missing.

This paper is organized as follows. Section II presents the related work. Section III presents the hardware and software architectures as well as implementation details. The experimental results are presented in Section IV. Finally, Section V concludes this work and raises possible future directions.

II. Related Work

There are currently different approaches to infer driver's attention as a consequence of the number of existing distraction sources. These sources can be detected by physiological, behavioral, and/or visual changes. Approaches based on physiological changes, on the one hand, focus on the driver to observe data such as heartbeat, brain activity, and eye condition to detect drowsiness. Approaches based on behavioral changes, on the other hand, focus on the vehicle and, as a consequence, uses the data from the vehicle such, as braking, acceleration, and sharp turns to detect user's inattention moments. Finally, approaches based on visual changes focus on information from the interior and exterior of the vehicle obtained with cameras and, as a consequence, can identify smartphone use, drivers with off-road gazing direction, traffic light running, and off-lane vehicle detection, for example.

This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. It was also supported by CNPq, FAPERJ Grants E-26/211.144/2019 and E-26/202.689/2018, and FAPESP Grant 15/24494-8.

Wang et al. [3] use data from electroencephalography during the operation of the vehicle to detect whether the driver is performing a task other than driving and, therefore, is distracted. This is an example of a physiological study for detecting drivers' attention. The approach has an accuracy of 90%, but it requires on-body devices, which may find resistance among users. As an example of a behavioral approach, there is the application proposed by Khanda-kar et al. [4]. This application monitors vehicle data and controls the use of smartphones. The system monitors driver's behavior such as smartphone use, braking, and abrupt lane changes. This system, however, only monitors and restricts the use of the smartphone and does not alert the user in case of distraction. Furthermore, Khanda-kar et al. use equipment in compliance with the OBD-2 standard to obtain data and vehicle diagnosis. In Brazil, for instance, this type of standard may be only partially available as the national standard, OBDBr-2, is taking place.

Today, there are many approaches to estimate gaze direction using drivers' head orientation [5]. This is a known problem from computer vision with several applications in different areas. For example, in psychological analysis [6], the head pose and movement can be used to infer patient depression; while in industrial scenarios, the head pose of a construction equipment operator can be used to identify blind spots and consequently avoid struck-by accidents [7]. Even though the utilization of eye tracking is intuitive for gaze direction detection, Lex Fridman et al. [8] show that the improvement over a head-orientation-based system is on average, of 5.4%. As a consequence, the complexity added to the system may not justify its implementation. The system proposed by Lamia and Hoque [9] uses head and eye orientation to determine the driver's attention. This is then an example of the visual approach. The system classifies the user as distracted if the angle of the driver's gaze direction exceeds a fixed threshold for a certain interval of time. In this case, the authors achieve 92% accuracy for detecting distraction. This approach, although efficient, proposes a fixed threshold, which may not be suitable for dynamic scenarios.

Craye et al. [10] propose a system mixing physiological, behavioral, and visual approaches. It uses heart rate, pedal position and steering wheel, and audio and video to determine the driver's attention in a simulator. Therefore, it covers all source, physiological, behavioral, and visual. Another interesting feature is the choice of a modular architecture, which uses three independent modules, visual, auditory, and signal. With this hybrid system, the authors could achieve a precision for detecting fatigue and distraction of 98.4% and 90.5%, respectively. Despite the remarkable precision, the system depends on invasive sensors, which may not be comfortable for drivers (oximeter CMS-50E). In addition, the system does not analyze distraction and fatigue in real-time. The authors explain that they have left this feature for future work.

In this paper, we are particularly interested in smartphone utilization as an important source of distraction. Smartphones contribute to the number of casualties in vehicular scenarios as drivers interact more and more with applications while driving [2]. Our proposal fits into the visual, behavioral, and physiological approaches, becoming a hybrid system. The inference of attention is made in real-time so that it is possible to alert distracted drivers. Also, the visual analysis is complemented by behavioral data, such as vehicle speed and acceleration.

III. System Architecture and Implementation

In this section, we present the hardware and the software architecture of the proposed system and the corresponding implementation details.

A. Hardware architecture

The proposed system is mainly composed of a set of input sensors, including a fixed camera, and a processing unit with enough memory. This processing unit must have enough computational power to process images in real-time. This is no longer an issue as more and more electronic devices are being embedded in vehicles. Also, additional information from the vehicle must be obtained from embedded or external sensors, e.g., location and speed from GPS and acceleration. The system output is a log file, an HTML file containing the vehicle trajectory, and, if desired, a video file with frames processed by the system. Therefore, the system must have persistent storage to save files for offline analysis. The use of a screen to watch the video output is optional.

Another requirement is an energy power supply that can be found within the car. Figure 1 illustrates the proposed system. In our prototype, the data is collected using a webcam and a Raspberry Pi, both connected to a laptop. The Rasperry Pi has an MPU-6050 accelerometer and a GPS u-blox EVK-5 connected. We use USB communication with the GPS and the I2C (Inter-Integrated Circuit) protocol with the accelerometer.

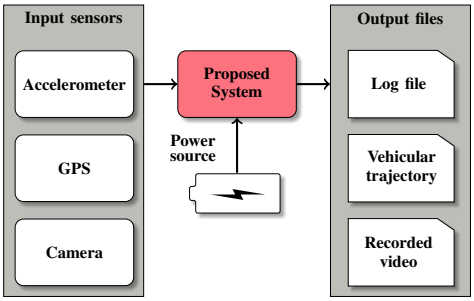


Fig. 1: Proposed system overview with a power source, a set of input sensors, and the produced files as output.

B. Software architecture

The system design is based on a modular architecture for maintenance, debugging, reuse, and update simplicity. Therefore, the system programming is separated into self-contained modules that implement different functionalities. The system contains a total of nine modules: the Main Module and the auxiliary modules called Video, Facial, Severity, Reception, Publication, Log, Notification, and Alert Module. Because the Log, Video, and Severity modules are asynchronous, they

are implemented as different threads from the main one. The system has also configuration files for user customization.

Figure 2 illustrates the relationship between system modules, where configurable modules have a gear figure at the top right. Also, different colors denote the corresponding threads in which each module is instantiated. Dashed arrows denote communication between modules installed at different devices, whereas continuous arrows denote communication involving modules implemented at the same device. The individual description of each module is presented next.

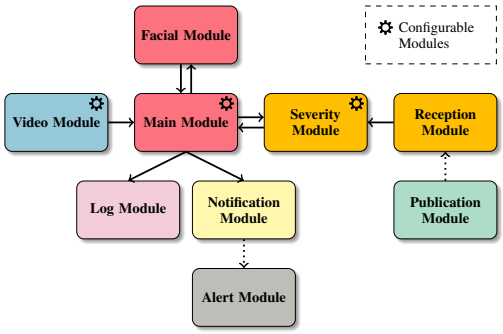


Fig. 2: Relationship diagram between system modules.

Main Module: The main function of this module is the orchestration of all auxiliary ones. Therefore, it runs the main program flow, transfers information from one module to another, as shown in Figure 2, and initializes the system. Using a configuration file, the user determines whether the system output will be persistently stored. If so, the Main Module records information such as head orientation, vehicle speed, and video frames. At the end of the system execution, the main module traces the vehicle trajectory on a digital map and records the result in an HTML file for later visualization.

The workflow conducted by the Main Module is the following. It first receives a frame from the Video Module and then conducts face recognition using facial landmark points obtained with the `Dlib` library [11]. Yet, the Main Module obtains thresholds regarding maximum head rotation angles from the Severity Module, which also considers data from additional input sensors. All the obtained data is sent to the Facial Module to have the estimated driver's head orientation and the driver's attention state received back. The Main Module sends to the Log Module all the received data from the additional input sensors and the information regarding the driver's head orientation. Finally, it sends as well the driver's attention state to the Notification Module.

Facial Module: This module estimates the driver's head orientation. To this end, the module receives facial landmark points extracted from video frames to compute the corresponding head rotation angles. If the estimated driver's head orientation continuously exceeds the maximum rotation angles, as determined by the Severity Module, the driver is considered distracted. The Facial Module also implements driver's blind eye detection using the EAR (Eye Aspect Ratio) metric [12], which also receives landmark points from video frames. Hence, we assign a distracted state to a driver who has blind eyes or is not looking ahead. This state is obtained by the Main Module and sent to the Alert Module using the Notification Module.

In more detail, the Facial Module receives from the Severity Module the facial landmark points. After that, it computes the rotation matrix, the EAR, and the rotation angles according to the longitudinal (yaw) and transverse (pitch) axes. The Severity Module uses these parameters to update implemented counters that are also used to infer distraction (explained shortly). The Main Module can then obtain the head orientation, eyes status, and the driver state (distracted or not) from the Facial Module using public methods.

As mentioned, the angles of interest for the system are those computed considering the driver's head rotation according to yaw and pitch axes. The sagittal axis is not used as the corresponding rotation is not considered a distraction (gaze direction can still be ahead). To compute the head pose, three coordinate systems are used: space, camera, and image coordinates. Computing rotation and translation, it is possible to transform a point in the 3D space into a point in the camera coordinate system. These points are then projected onto the image coordinate system, which is obtained with `Dlib`, using camera configuration parameters. Among the camera configuration parameters, we have the camera focal distance, the image optical center, and the radial distortion parameters, which are approximated herein to avoid camera calibration.

To obtain the landmark points from the driver's face at the space coordinate system, we use a generic 3D model of a human face, available to download at the TurboSquid website [13]. The nose is considered the system origin, the points of interest are then identified, and their coordinates are manually obtained. The process to acquire the coordinates is conducted using the 3D Blender software. Thus, the points obtained from the space coordinate system, the points computed by `Dlib` and the camera parameters allow the computation of the driver's head rotation angles using the Levenberg-Marquardt optimization method [14]. This method returns the rotation and the translation vectors. The function `Rodrigues` from the `OpenCV` library is then used to convert the rotation vector into the desired rotation matrix.

Driver's distraction is computed considering the head direction angles and the average EAR value of both eyes. If the rotation angles are above predefined thresholds or if the EAR is below a predefined threshold after a certain number of consecutive frames, the driver is considered distracted. The EAR is computed using six landmark points from each driver's eye. The number of consecutive frames a driver is considered distracted is controlled using counters, one for each rotation angle and another for the blind eyes. We use a sliding window approach that updates all the proposed counters whenever a new video frame arrives. Hence, a driver is considered distracted if one of these counters reaches a value above the corresponding tolerated threshold. It is worth mentioning, that these counters are decremented at each frame the driver is

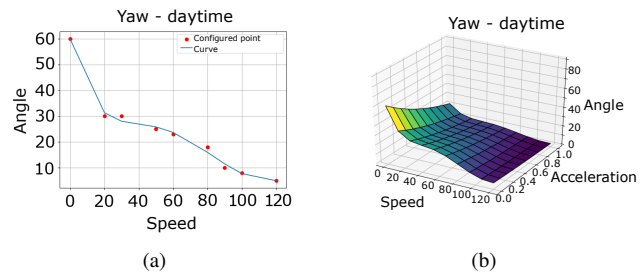
attentive. We understand that alerts must not be triggered on a per-frame basis to smooth the system reaction and avoid false alerts, which would be cumbersome for the driver. The thresholds are predefined at the Severity Module and can be configured according to the driver's preferences.

Video Module: The system can analyze previously prerecorded videos in addition to the real-time video streaming from the camera. This additional functionality is implemented for offline analysis of videos, e.g., for auditions. Thus, the system can infer driver attention using a video recorded by a different source or a video recorded by the system itself. The Video Module determines the video input using a configurable parameter.

The `OpenCV` library is used to capture the video input as well. Each option, prerecorded video or real-time streaming, requires a different implementation. For the former one, the `OpenCV` library blocks the system I/O, which is implemented as a separated thread using a buffer to avoid negative impact on the system performance. Instead of reading from the file, the frame is read from the buffer. For the latter option, the frames are directly obtained from the camera.

Severity Module: The Severity Module correlates the user's behavior (collected from the additional input sensors) with his attention when driving the vehicle. It can be understood, therefore, that the Severity Module is responsible for the system intelligence. For example, when driving at low speed, a driver has a maximum allowed angle for head rotation greater than the angle of another driver at a higher speed. As the speed increases, the permitted angle decreases because the driver must be more attentive. The parameters used by the Facial Module are generated from the Severity Module. Therefore, this module has as its main functionality the contextualization of the driver's attention status, using data from the Reception Module, and the definition of tolerable thresholds used by the Face Module. Also, this module implements the counters explained at the Facial Module.

The Severity Module computes the thresholds used for the driver's head rotation angle; one threshold for the yaw and another for the pitch axis, at its initialization. In addition, we consider that the threshold is different during the day and at night. Hence, there are four thresholds in total. These thresholds are not computed as a single value, but as curves considering different speeds and accelerations. We first plot 2D curves taking into account a 5th-degree polynomial regression model to fit different (speed, angle) points predetermined at the configuration file. Hence, we adopt an exponential decay empirically computed at preliminary tests to also consider the acceleration dimension. Each 3D surface then defines the maximum angle a driver can rotate its head considering a single axis and one period of the day. We assume that at night, with less luminosity, it is more important to be more conservative and, therefore, use smaller thresholds. Also, we use pre-configured points, as they do not need previous training and can be customized for each driver. Figures 3(a) and 3(b) illustrate, respectively, the curve obtained for the yaw axis during the day and the corresponding surface after considering the acceleration dimension.



(a) (b)

Fig. 3: Head rotation threshold at the yaw axis during the day. (a) Curve considering the vehicular speed. (b) Surface also considering the vehicular acceleration.

The Severity Module initializes an instance of the Reception Module to request updated data from the input sensors. These data are obtained using periodic remote procedure calls with an interval also predefined at the configuration file. Finally, the module uses the `skyfield` library to determine if this is day or night. The thresholds are sent to the Main Module. For resilience, if the connection to a sensor fails, standard values are used.

Reception Module: This module is instantiated by the Severity Module and its goal is to receive messages containing data coming from the additional input sensors. To accomplish that, it executes remote procedure calls to obtain the data from the input sensors. The data payload received is forwarded to the Severity Module, which further computes the thresholds for the driver's head rotation angle. The data payload contains a five-tuple composed of ⟨`latitude; longitude; speed; y-axis acceleration; z-axis acceleration`⟩. By simply changing the tuple, this implementation permits the addition of new input sensors without affecting the system. Also, the communication is conducted in separate threads, which does not affect the main system performance.

We use asynchronous communication using a pub/sub approach. Hence, the Reception Module consumes the data produced by the input sensors using message queues, one to request and another to receive the corresponding data. The proposed system uses the RabbitMQ, which provides an API containing procedures for remote procedure calls. Also, the implementation of a new input sensor (i.e., a new publisher) requires only a new publisher and a new queue. The Reception Module has a timeout parameter to cancel pending calls. Thus, if the RabbitMQ or the Publication Module are unavailable, a timeout exception is triggered to be further handled by the Severity Module.

Publication Module: This module is the interface between the system and the additional sensors. The Publication Module is responsible to receive the data from the input sensors and create the message to be sent to the Reception Module. This

module, in our experiments, obtains the data from the GPS and the accelerometer.

Log Module: This module records the vehicle status, such as speed, position, and acceleration, received from the Main Module. In addition, it registers the driver's status, such as inferred head orientation, EAR, and whether the driver is considered distracted. The log is periodically written taking into account a configurable time interval for offline analysis. In addition, the Log Module is implemented asynchronously in a different thread so as not to impact the main system performance.

Notification Module: This module sends the driver's status, distracted or attentive, to the Alert Module. To accomplish that, the implementation follows also a pub/sub approach. The Notification module publishes the driver's status to be consumed by the Alert Module. This, besides being asynchronous, simplifies the addition of new modules, such as one that could kill an application in the driver's smartphone.

Alert Module: This module receives the driver's status from the Notification Module. If the driver is distracted, this module must give an alert. The importance of the alert is to keep drivers informed about their driving conditions and, consequently, to become more attentive to possible accidents. Whenever the driver is considered distracted, a led is turned on and a beep sound is emitted. This module runs inside the Raspberry Pi, independently from the Publication Module, consuming driver's information from the Notification Module.

IV. Experimental Results

We first conduct preliminary experiments to validate the system operation in a controlled scenario and then we conduct experiments in a real scenario.

A. Preliminary results

This experiment first evaluates the Facial Module without sensors and then with the support of additional input sensors.

Without input sensors: We start these experiments using default threshold values for the driver's head rotation angle. Figure 4 shows the Facial Module operation, without additional sensors. We use as default values for the yaw and pitch axis threshold, respectively, $25°$ and $5°$. When these limits are reached, the module infers that the driver is possibly distracted. In Figure 4(a), the system could infer that the driver is possibly distracted as he is looking right with an angle of $34.98°$, while in Figure 4(b) the system could infer that the driver is possibly distracted as he is looking down with an angle of $-15.80°$. It is worth mentioning that positive and negative angles denote, respectively, up or down head direction or right or left. The comparison to the thresholds must consider absolute values.

Figure 5 shows that the current driver status, distracted and not distracted, changes along the time according to events on the yaw and pitch axis. Note that the system infers that the driver is distracted sometime after the system goes above the yaw superior limit. Similarly, the same behavior happens when the system moves below the yaw superior limit and, afterward, when it goes below the pitch inferior limit. This happens because the proposed system considers counters, used to avoid premature conclusions.

(a) Looking right (EAR = 0.31, yaw = $34.98°$, pitch = $4.61°$).

(b) Looking down (EAR = 0.28, yaw = $-20.65°$, pitch = $-15.80°$).

Fig. 4: Facial Module evaluation using default values.

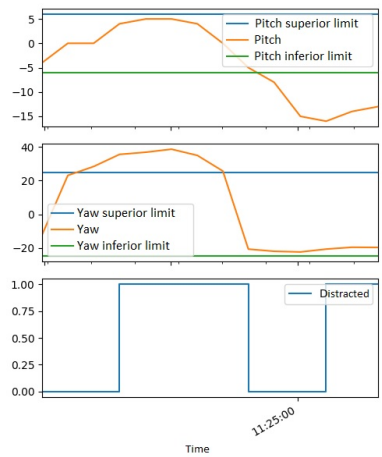
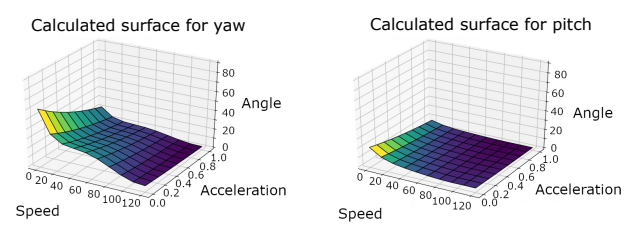


Fig. 5: Driver's distraction detection using pitch and yaw axis.

With input sensors: Considering the input sensors, the pitch and yaw thresholds must also vary as a function of the speed and the acceleration. In this experiment, the speed is increased by 5 km/h and the acceleration by 0.1 m/s^2 per second. Figures 6(a) and 6(b) show the surface generated to compute the maximum head direction rotation at the yaw and pitch axis using pre-configured points at the configuration file.



(a) Surface that delimits yaw axis rotation.

(b) Surface that delimits pitch axis rotation.

Fig. 6: Surfaces used in the experiment using data from vehicular speed and acceleration.

Figure7 shows that the thresholds for the driver's head direction angles are influenced by the speed and the acceleration.

The thresholds quickly decrease considering absolute values, as the speed and the acceleration increase. The distraction is then inferred sometime after the yaw superior limit is overtaken at near 12:39:31 PM. There is a clear tradeoff between system reaction and possible false alarms that can be further investigated.

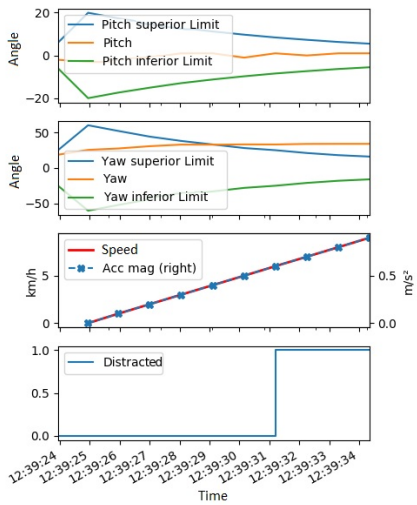
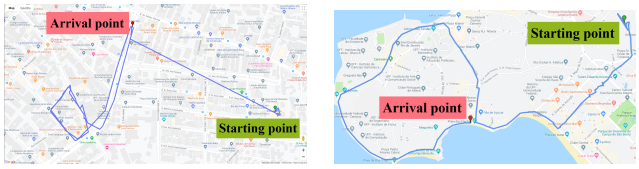


Fig. 7: Driver's distraction detection upon acceleration and speed changes.

B. Real scenario results

We have conducted two experiments after installing the prototype in a vehicle. Figures 8(a) and 8(b) show the maps generated by the system after each experiment.



(a) Afternoon trajectory. (b) Evening trajectory.

Fig. 8: Trajectories recorded by the system, starting at the green point and arriving at the red point.

Figure 9 shows the obtained results for head rotation according to the pitch and yaw axis. Whenever the head direction angle exceeds the predetermined thresholds, the system alerts the driver about the distraction after a certain number of consecutive frames. Note that when the system fails to obtain the GPS speed, as observed in Figure 9, the default values are used.

Regarding blind eye detection, Figure 10 shows the obtained results. Whenever the EAR falls below the pre-defined threshold, the system can also system alert the driver about the distraction. We use a pre-defined value for EAR threshold equal to 0.2 in our experiments. Observe that the blind eye detection infers that eyes are closed if the EAR value is below

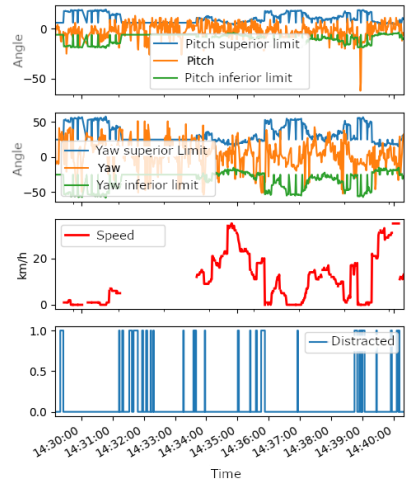


Fig. 9: Driver's distraction detection in real scenarios.

the threshold after a certain number of consecutive frames. We also use a particular counter to avoid consecutive alerts and false alarms.

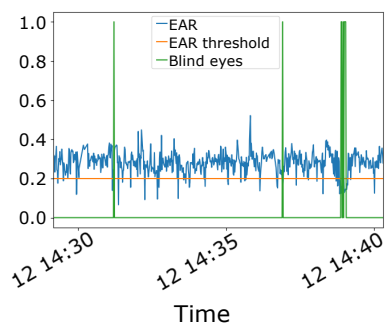


Fig. 10: Blind eye detection using EAR.

Figures 11(a) and 11(b) illustrate distraction periods from the driver, which are correctly captured by the system. In Figure 11(a), the driver is looking right with an angle above the maximum yaw threshold, dynamically computed the system, $27.41° > 19.48°$. In Figure 11(b), the driver has the eyes closed with an EAR of 0.14. The system can similarly infer that the driver is distracted with closed eyes.

We raise some possible points for improvements, which may happen when the driver looks right or left for a long curve, as illustrated in Figure 12. This, however, can be avoided by changing the values set for the proposed counters or the predetermined points used by the regression model. This figure also shows that the system continues operating with default values for yaw and pitch thresholds for maximum head direction rotations, even after losing the GPS signal. The system, however, may not have a desirable performance when the driver rotates the head until the landmark points are not be detected. In this case, the system could not infer the driver's distraction for a certain number of frames, as illustrated in Figure 13(a). This problem happens more often at night, as

(a) Head direction looking right (EAR = 0.29, yaw = 27.41°, pitch = 5.83°).

(b) Blind eye (EAR = 0.14, yaw = −9.38°, pitch = −12.48°, max yaw = −3.40°, max yaw = 19.48°, max = 55.94°, max pitch = 18.61°).

Fig. 11: System detecting moments of driver distraction in the real scenario.

the low light intensity can degrade the system operation. With low light, the system has difficulties detecting the landmark points on the driver's face, as seen in Figure 13(b). We believe that this issue is easily fixed using infrared cameras, which will be our next step, left as future work.

Fig. 12: System detecting a false positive when the driver continuously look to right after a long curve (EAR = 0.29, yaw = 33.04°, pitch = 10.49°, max yaw = 25.00°, max pitch = 6.00°). Max yaw and max pitch are standard values given the GPS signal loss.

(a) Looking left with low light intensity.

(b) Extremely low light intensity.

Fig. 13: Failure to detect the driver's landmark points as a consequence of the low light intensity.

V. Conclusion

This paper proposed a real-time system to alert drivers about moments of distraction. The idea was to implement a lightweight modular system able to infer driver's distraction based on driver's head pose and blind eye detection. In this paper, we were particularly interested in inferring distraction when the driver looks down, to detect smartphone utilization. To infer head pose and eye blindness, we couple information from landmark points from the driver's face obtained by cameras with additional information from input sensors, such as GPS and accelerometer. We assume that drivers have less freedom to change head orientation at higher speeds and with higher vehicular acceleration. Our results in controlled environments and in real scenarios confirmed the system ability to detect driver's distraction. We identified, however, points for further investigation such as in scenarios with low light intensity.

Acknowledgment

This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. It was also supported by CNPq, FAPERJ Grants E-26/211.144/2019 and E-26/202.689/2018, and FAPESP Grant 15/24494-8.

References

[1] J. K. Caird, K. A. Johnston, C. R. Willness, M. Asbridge, and P. Steel, "A meta-analysis of the effects of texting on driving," *Accident Analysis & Prevention*, vol. 71, pp. 311–318, 2014.

[2] National Center for Statistics and Analysis, "Overview of motor vehicle crashes in 2019," DOT HS 813 060, U.S. Department of Transportation, Tech. Rep., 2019. [Online]. Available: https://www.nhtsa.gov/press-releases/nhtsa-releases-2019-crash-fatality-data

[3] Y.-K. Wang, S.-A. Chen, and C.-T. Lin, "An EEG-based brain–computer interface for dual task driving detection," *Neurocomputing*, vol. 129, pp. 85–93, 2014.

[4] A. Khandakar, M. E. Chowdhury, R. Ahmed, A. Dhib, M. Mohammed, N. A. Al-Emadi, and D. Michelson, "Portable system for monitoring and controlling driver behavior and the use of a mobile phone while driving," *Sensors*, vol. 19, no. 7, p. 1563, 2019.

[5] S. J. Lee, J. Jo, H. G. Jung, K. R. Park, and J. Kim, "Real-time gaze estimator based on driver's head orientation for forward collision warning system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 254–267, 2011.

[6] S. Alghowinem, R. Goecke, M. Wagner, G. Parkerx, and M. Breakspear, "Head pose and movement analysis as an indicator of depression," in *Humaine Association Conference on Affective Computing and Intelligent Interaction*, 2013, pp. 283–288.

[7] S. J. Ray and J. Teizer, "Coarse head pose estimation of construction equipment operators to formulate dynamic blind spots," *Advanced Engineering Informatics*, vol. 26, no. 1, pp. 117–130, 2012.

[8] L. Fridman, J. Lee, B. Reimer, and T. Victor, "'Owl' and 'Lizard': patterns of head pose and eye pose in driver gaze classification," *IET Computer Vision*, vol. 10, pp. 308–314(6), 2016.

[9] L. Alam and M. M. Hoque, "Real-time distraction detection based on driver's visual features," in *International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1–6.

[10] C. Craye, A. Rashwan, M. S. Kamel, and F. Karray, "A multi-modal driver fatigue and distraction assessment system," *International Journal of Intelligent Transportation Systems Research*, vol. 14, no. 3, pp. 173–194, 2016.

[11] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[12] T. Soukupova and J. Cech, "Eye blink detection using facial landmarks," in *Computer Vision Winter Workshop*, 2016.

[13] "Male head.obj," https://www.turbosquid.com/FullPreview/Index.cfm/ID/346686, accessed at 2019-08-14.

[14] S. Roweis, "Levenberg-Marquardt optimization," *Notes, University Of Toronto*, 1996.