Drowsiness Detection

A capstone project report submitted in partial fulfillment of the requirement for the degree of

Masters of Engineering

In the *Name of Department* Graduate Program,

College of Engineering & Applied Science

04/08/2019

Artur Lach

Abstract - Driver falling asleep behind the steering wheel is not a new a problem, but a problem that has been reason of many accidents [III]. Many detection, as well as prevention methods have been researched, proposed and some of them even implemented in vehicles. So far implementations of such systems in vehicle tend to be indirect and not necessary linked to the driver. One method that has been researched is to use a camera to monitor the driver and determine the drowsiness level of a driver by processing the video stream. Example of such system was developed and is described in the following report. Hardware of the implemented system consists of web cam, pair of speakers and 7” touchscreen display connected to Raspberry Pi single board computer. System works by retrieving frames from the video stream, processing them to detect driver’s face and eyes, analyzing detected eyes to determine if they are closed or opened, calculating eyes related data such as blink counter, displaying driver drowsiness level (determined based on the collected data), and warning driver with a warning sound if his/her eyes are determined to be closed longer than a threshold. Additionally, part of the goal of this project was to mimic how similar system could be implemented in an actual vehicle, and therefore data of application is displayed on the attached 7” display by using a simple graphical user interface. Similar application could be for example installed on the infotainment system of the vehicle. Although, the system is far from perfect and would definitely need improvements before being production ready, it shows that similar system could be installed in today’s vehicle which have enough of computing resources and power.

1. Introduction

Drowsy driving is a problem that most of the people experience at least once in a lifetime. Although one might feel great and rested before getting behind the wheel, that feeling might drastically and quite quickly change. Often the transition from being aware to a condition of micro-sleeping is so abrupt that person doesn't even realize it [III]. There are also times when people deliberately get behind the wheel while knowing they are already tired. Both cases frequently end, in the best case scenario with a close call, and in worst with an accident. According to NTHSA drowsy driving led to about 72,000 accidents in 2013 [IV]. There are two aspects of the problem that need addressing. First is a need to recognize if the driver has fallen asleep and then taking measures to wake him/her up, and second is an early recognition of driver getting tired and providing that information. Both academia and automotive industry have attempted to research and solve this problem, with the first focused more on how to recognize drowsiness level of the driver and the second on implementing preventative measures in the vehicles. One of the conducted researched for example utilized EOG and EEG signals to recognize driver blinking, which due to its intrusive nature was never a good candidate for an implementation in production systems. On the other hand auto manufacturers have already implemented some warnings and safety features that somewhat indirectly address driver drowsiness: for example they can display warning message suggesting driver take a break or vibrate the steering wheel if it is determined that the vehicle is crossing the lanes without driver's intent. Another method that has been researched and is a good candidate for the implementation by manufacturers is to use a driver facing camera to monitor the driver and with use of computer vision determine drowsiness of the driver. As a matter of fact such method is behind Subaru's "DriverFocus" features which is described as "driver monitoring system that uses facial recognition software to identify signs of driver fatigue or driver distraction" [VI]. Such system has also been implemented and is described in this report.

2. Methods

a) Hardware

Although, project is mainly based on software, which could have been installed and ran on regular, general use computer, it was chosen to utilize a hardware that somewhat closer represents computing modules found in vehicles. For that reason Raspberry Pi 3b+ was chosen as the brain of the system. Raspberry Pi is a single board computer that is very well known and often used for prototyping. It uses Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit SOC clocked at 1.4 GHz, Broadcom Videocore-IV GPU, 1GB LPDDR2 SDRAM. It also exposes multiple interfaces such as GPIO, USB, 3.5mm analogue audio-video jack, Ethernet, HDMI, DSI and has wireless capabilities. It runs an a full operating system, most often Linux Raspbian distribution, from an attached SD card. For the purpose of this project Raspberry Pi is interfaced with following components:

- web cam via USB connection
- pair of speakers via analogue audio-video jack and powered via USB connection
- 7" touchscreen display via DSI connection

These components together are meant to loosely represent what could be found in an actual vehicle: for example infotainment module with a display and internal camera. However additionally by using these components proposed system could also be installed in the vehicle as an add on.

b) Software

Software is the main part of the drowsiness system. This section of the report describes both architecture of the system as well tools and technologies used.

I. Tools and technologies

Although, embedded systems found in production are typically programmed in C/C++, many prototypes, especially ones dealing with computer vision and machine learnig are instead developed using Python. Being an interpreted language, Python runs slower than compiled C/C++, however, thanks to abundant number of libraries and general ease of use Python is a great choice for such prototypes. Due to those benefits Python3 was selected as development language for the project. The application also uses and therefore has dependencies on multiple Python libraries. Some of the libraries are actually installed by default with Python installation while others need to be downloaded and installed. Below is a list of the most important libraries used in the project and their short descriptions:

- OpenCV - defacto a standard library for any computer vision work
- Dlib - popular machine learning library
- Imutils - helper library that makes OpenCV easier to work with
- Scipy - open source library used for mathematical operations
- Numpy - popular scientific computing package
- Tkinter - graphical user interface development library
- Pygame - typically used for developing games, but it can be quite easily used for sound playback

Since Python is not a compiled language it does not create executables applications that can simply be loaded to memory and ran. All Python applications need to be run via interpreter which requires explicitly calling it out to start the program, ex. python test_script.py. Since, it was desired to make the application easy to run shell scripting was utilized to trigger execution of the application via the interpreter.

II. Architecture

Figure 1 below presents use cases associated with drowsiness detection system.

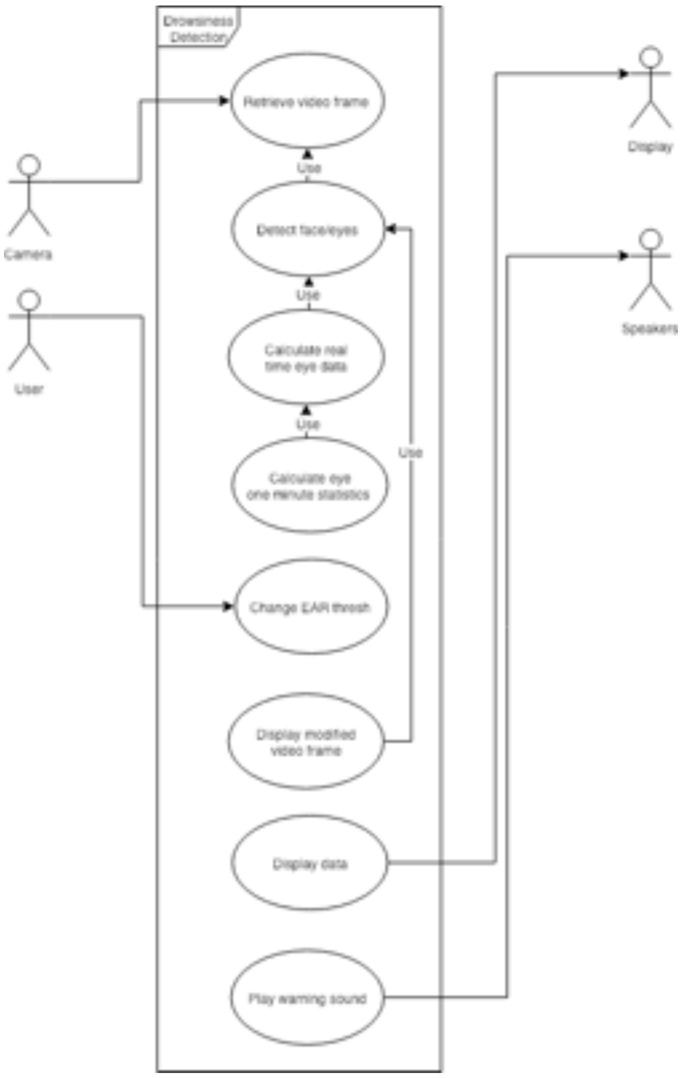


Figure1: Use Cases

Although there are eight use cases there are really four specific things that the drowsiness application is meant to achieve: first retrieve, then process and analyze the image, and finally display the data back to the user. These steps map very well into general concept of model, view, controller architecture, where controller is responsible for interfacing with the camera and speakers, model is responsible for processing frames and analyzing data retrieved from them, and view is used to display the information. Software is developed in object oriented programming manner with functionality split into following classes:

- WarningAnnouncer - used to playback warning sound via connected speakers
- DrowsAnalyst - used to process the image frame to determine and calculate eye data, such as blink counter, maximum time of eyes closed, user sleeping and others
- SleepDetectorApp - this is a main class o the application, that utilizes classes listed above as well as classes from other libraries to build graphical user interface, retrieve frames from the camera and provide it for analysis, trigger warning sound when user has fallen asleep, and update graphical user interface with latest data

Flow of events of the application is very sequential and is presented with flow chart in Figure 2 below.

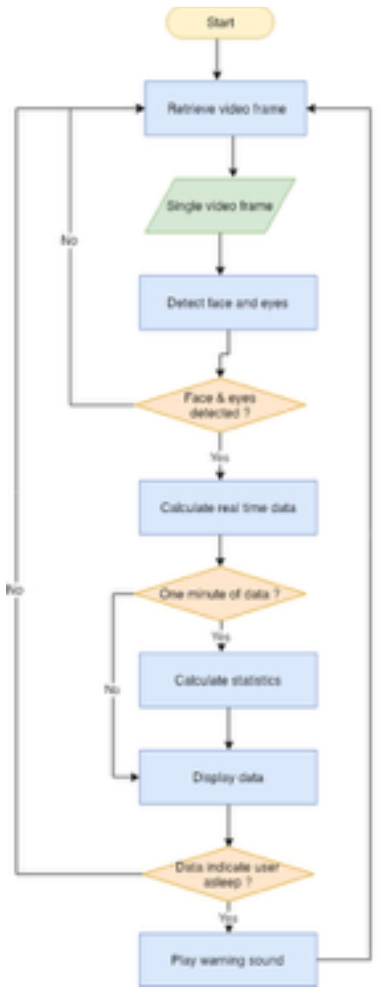


Figure 2: Events flow

Main loop of the application is programmed to repeat steps shown above every 10 ms [IX]. However, how often new video frame is fetched exactly is depended on amount of time it takes to process it. It is possible that it takes longer than 10 ms before new frame is retrieved. How much longer has not been measured, since it has been decided that small variations should not have a negative impact on the application itself, since typical blink time of a person is around 100 ms or more. The major part of drowsiness detection application is an algorithm that processes the image. The result of this processing is short and long term eye data, which is then used for displaying it to the user, determination if driver has fallen asleep and determination of driver’s general drowsiness level. Collected eye data includes: eye aspect ratio (EAR), continuous time that eyes have been closed (ETC), continuous time that eyes have been opened (ETO), number of blinks in last minute (BPM), difference between current and previous number of blinks in a minute (BDPM), combined time of eyes closed in last minute (ATCPM), difference between current and previous combined time of eyes closed (DATCPM), maximum time that eyes have been continuously closed in last minute (MAXTCPM), minimum time that eyes have been continuously opened in last minute (MINTCPM) and drowsiness level. The algorithm works by first detecting a face in provided video frame by using cascade classifier provided by OpenCV library [V]. If face is detected the classifier returns its coordinates in the frame. Returned coordinates are then used to create a rectangle around detected face, and passed into Dlibs shape predictor. Shape predictor returns coordinates of major facial landmarks such as eyes, nose, mouth etc. In the next step eyes coordinates are used to calculate eye aspect ratio, which is simply ratio of distances between the vertical eye landmarks and the distances between the horizontal eye landmarks [II, VIII]. Based on the paper by Soukupová and Čech’s eyes aspect ratio can be accurately used to determine if eyes are closed or not [X]. The ratio is higher when eyes are opened and decreases toward when eyes are closed. By comparing eye aspect ratio, calculated from the received frame, to a set threshold, algorithm decides if driver’s eyes are closed or opened. Ear threshold is by default set to 0.22, but can be modified by providing new value to the input box on the user interface. Once state of the eyes in current frame is determined it can then be combined with their state in the previous frame to deduce additional statistics mentioned previously including the most important one which is a drowsiness level. There are four possible drowsiness level states: low, medium, high and extreme. Determination of drowsiness level is performed according to the state machine shown in Figure 3 below.

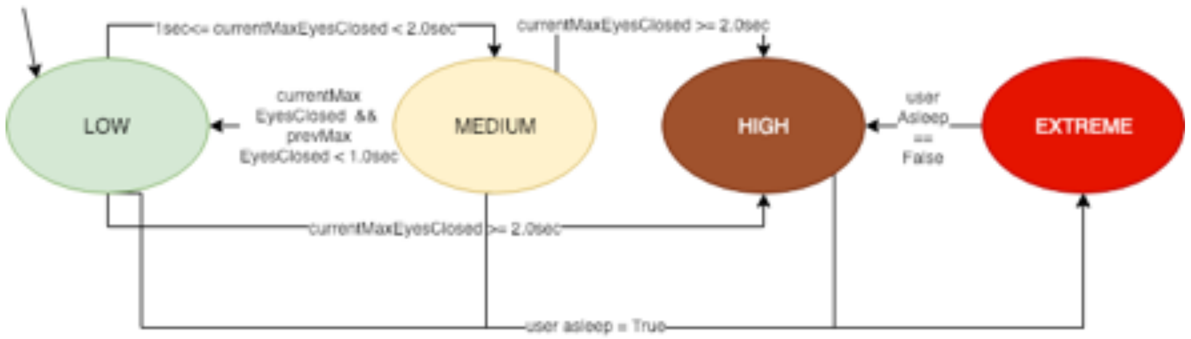

1sec<= currentMaxEyesClosed < 2.0sec
currentMaxEyesClosed >= 2.0sec
LOW
currentMax EyesClosed && prevMax EyesClosed < 1.0sec
MEDIUM
HIGH
user Asleep == False
EXTREME
currentMaxEyesClosed >= 2.0sec
user asleep = True

Figure 3: Drowsiness level state machine

Important to note is that user asleep condition shown in the state machine is true whenever the driver has had his/her eyes closed for longer than 3 seconds. Determination of user asleep is always completed before executing drowsiness level state machine which in turn guarantees that if the driver is currently sleeping the drowsiness level will be displayed as extreme and accompanied by loud warning sound.

3. Results

Drowsiness detection system was successfully developed and its operation is demonstrated using screenshots shown in the Appendix A. Additionally, all necessary files to run the application as well as short videos presenting its operation can be found inside of attached DrowsinessDetection directory. Assuming that all dependencies are installed the application can be started by executing launch_drows_detector.sh script. Once the application is running it can be closed by pressing “esc” key. It is also possible to make the application run right after Raspberry Pi boots up (without displaying the desktop) [VII] by first executing bootWithApp.sh script inside DrowsinessDetection/StartupLaunch/WithApp/ by the user with administrative privileges (ex.

sudo), and then rebooting the system. If running on boot functionality has been enabled and is later not desired anymore it can be reverted by executing bootWithoutApp inside Drowsiness-Detection/StartupLaunch/WithoutApp/, once again with administrative privileges.

When application is running, data on the left side is “real time” (it is not exactly real time since it is updates at best every 10 ms as described in previous section) and on the right are calculated statistics. Real time data section of the GUI also provides and an input box for setting the EAR threshold. Since it was noted EAR changes slightly depending on position of the camera and the user in the frame, this input box was added so that the system can be fine tuned to avoid false blink detections. In theory EAR value should stay consistent between eyes closed and eyes opened however it always fluctuates even if the state of the eyes didn’t change. As long as EAR value does not fall below EAR threshold the BPM should not be incremented. ETC and ETO values indicate time of eyes closed and eyes open respectively. The values are constantly incremented depending on user having his/her eyes closed. Seeing ETC change is difficult since it only changes when eyes are closed. BPM is updated every time blink is detected, and is a simple counter. Blinks are counted for 1 minute, and then restarted from 0. DBPM is updated every minute and displays the difference between number of blinks counted over last minute and a minute before then. ATCPM is a sum of time user has his/her eyes closed over the last minute. The value is updated after every blink and is reset to 0 after a minute. Similarly to DBPM, ATCPM is a delta of current and previous values, but for ATCPM. MAXTCPM and MINTCPM represent maximum and minimum time eyes have been closed in the last minute. After every blink, time of eyes closed is compared to current maximum and minimum stored, and updated if appropriate. Just like other statistics data, MAXTCPM and MINTCPM are reset after one minute. The horizontal bar on the bottom indicates current drowsiness level. The color of the bar changes according to the drowsiness level: green for low, orange for medium, brown for high and red for extreme. Extreme state indicates system detected user has fallen asleep and therefore warning sound playback starts in an attempt to wake the user up. Video stream is displayed in the middle of the user interface. This is done mostly for debugging purposes in order to identify if the user’s face and eyes were detected in the stream. Red outlines around the eyes indicate that face and eyes detections were successful. If the face is not properly detected in the stream then all displayed data is 0 and the video stream is unmodified.

4. Discussion

Development of drowsiness detection system was an interesting project that required usage of software development, some knowledge of intelligent systems and plenty of research. Important to highlight is the fact that the focus of this project was not a development of new face or eye blink detection algorithms, but rather a showcase of the end product. Although, the presented system could in theory be installed in the vehicle as is, it serves a better purpose as a demonstration of what could be integrated into existing vehicle technology, and as mentioned previously Subaru has already developed such system. Presented drowsiness system is definitely not perfect and would require improvements before making it into production ready system. In production implementation video stream of the driver would most likely not be displayed on the infotainment screen, since it would be a distraction more than being helpful. Also, the algorithm used for determination of drowsiness level would most likely be made much more sophisticated. Current implementation inspects only one of the collected data points, a better system could not only fuse and consider multiple data points, but also apply machine learning technology to better analyze the data. Additionally, during testing it was found that EAR was somewhat dependent on the position of the camera and user in the video frame. This for example led to false blink detections when the set EAR threshold was not appropriately adjusted. The workaround solution of allowing EAT threshold adjustments by the user, simply would not be an option in production system. As a matter of fact both face and eyes detection would require additional work in order to make it more consistent and reliable.

References

I. Anzalone, L. (2018, October 10). Training alternative Dlib Shape Predictor models using Python. Retrieved from https://medium.com/datadriveninvestor/training-alternative-dlib-shape-predictor-models-using-python-d1d8f8bd9f5c

II. Amaratunga, T. (2016, October 29). Getting Dlib Face Landmark Detection working with OpenCV. Retrieved from https://www.codesofinterest.com/2016/10/getting-dlib-face-land-mark-detection.html

III. Claiborne, R. (2012, December 3). 'In the Blink of an Eye': Dozing While Driving. Retrieved from https://abcnews.go.com/Technology/blink-eye-dozing-driving/story?id=17870880

IV. Drowsy Driving Guide: Risks and Prevention. (2017, April 12). Retrieved from https://www.-tuck.com/drowsy-driving/

V. Haar Feature-based Cascade Classifier for Object Detection¶. (n.d.). Retrieved from https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#cascadeclassifier-detectmultiscale

VI. Hawkins, A. J. (2018, March 28). Subaru will use facial recognition technology to detect driver fatigue. Retrieved from https://www.theverge.com/2018/3/28/17173358/subaru-forester-facial-recognition-ny-auto-show-2018

VII. Jhalani, R. (2019, February 26). Execute Script At Startup On The Raspberry Pi. Retrieved from http://pythonchoice.com/execute-script-at-startup-on-the-raspberry-pi/

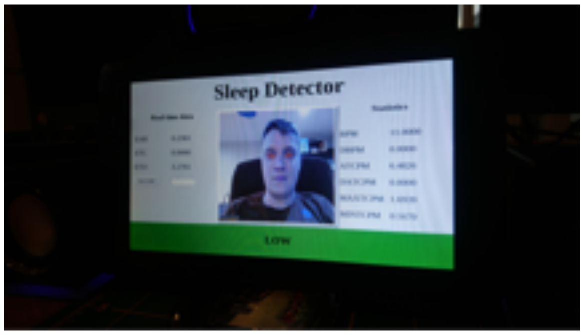
VIII. Rosebrock, A. (2017, April 24). Eye blink detection with OpenCV, Python, and dlib. Retrieved from https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/

IX. Silisteanu, P. (2018, April 21). Python OpenCV - show a video in a Tkinter window. Retrieved from https://solarianprogrammer.com/2018/04/21/python-opencv-show-video-tkinter-window/

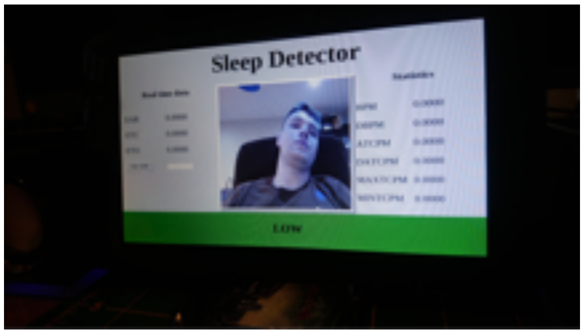
X. Soukupova, T., & Cech, J. (2016, February 3). Real-Time Eye Blink Detection using Facial Landmarks (Tech.). Retrieved http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

Appendix A

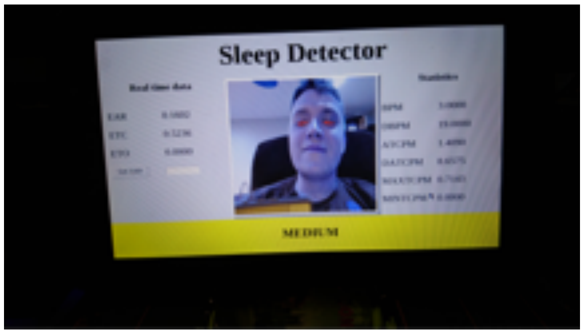
System setup



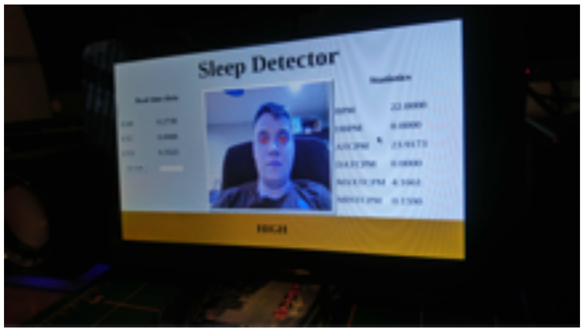
Lows drowsiness level



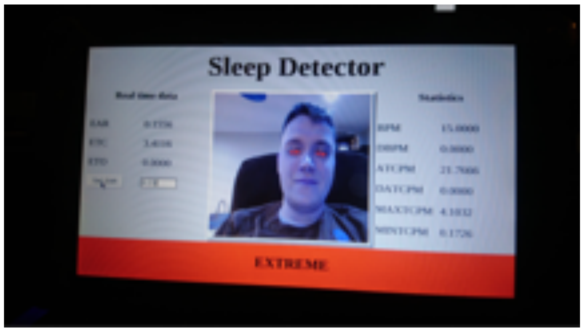
Eyes not detected in the stream



Medium drowsiness level



High drowsiness level



Extreme drowsiness level/user asleep