Detecting Driver Weariness Using Cameras

Daemon Macklin
Department of Maths and Computing
Waterford Institute of Technology
Waterford, Ireland
20075689@mail.wit.ie

***Abstract*— this document will outline the design and implementation of a Driver Monitoring System (DMS). I will be investigating two methods of detecting the eye states of a driver. And then implementing the superior method into a DMS.**

Keywords - Matlab, Classification, SVM, Eye, Sobel, Hough, ADAS

I. Introduction

Driver Monitoring is an ADAS feature where a number of metrics about the driver are constantly monitored by the vehicle. Metrics such as heart rate and acceleration/braking speed can be used to monitor if the driver is driving safely. With the advent of image processing it is possible to monitor the facial expressions of the driver. Namely if the driver is asleep or awake. The aim of this project is to use matlab to tell if the user is asleep or awake based on an image/video feed

II. Theory

The Driver Monitoring System will use a camera pointed at the driver. It will be relatively easy to get a good image of the driver as the driver seat is a fairly consistent environment. The images from the camera will then be processed to gather data on the eyes of the driver. This data will then be input into a model which attempts to predict if the drivers eyes are open or closed. The next problem is how does the computer know that the driver has their eyes open.

III. Classification

A. Data Gathering

In order to generate an accurate model that will work in a large number of situations we need to get as much data as possible. Thankfully, a lot of work has been done in the field of facial recognition, and as a result a lot of different datasets exist.

For this investigation I used two different data sets. The first is the Labeled Faces in the Wild(Vis-cs.umass.edu, 2019) Data set. This was made up of about 13,000 images, and I used some of these images for the open eyes data. For the closed eyes data I used the Closed Eyes in the Wild(Parnec.nuaa.edu.cn, 2019) Data set. This was made up of 2423 images.

B. Image Processing

The model is going to use the Eigen Features(Yao Hongxun et al., n.d.). This technique is used in computer vision to pick out and score different features of a face. In this case we are only interested in getting the Eigen Features around the eyes and build a model to tell the difference between features of closed eyes vs open eyes. The vision toolbox in matlab comes equipped with the CascadeObjectDetector function which uses the Viola-Jones algorithm to detect faces. This function can be specified to detect where on a face the eyes are located, as seen in Fig. 1.

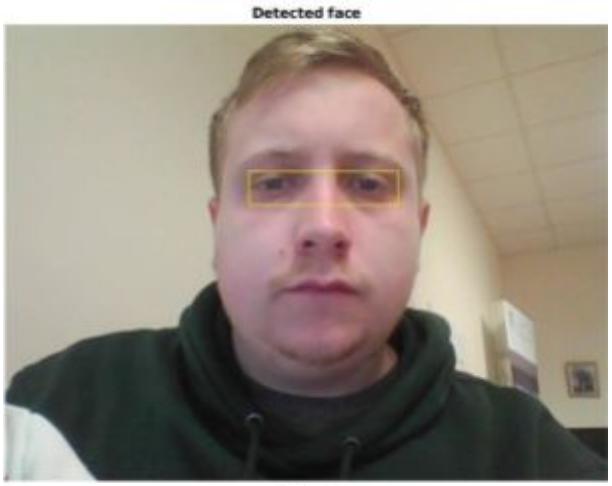

Fig. 1

Onces the eyes have been located we can use the matlab detectMinEigenFeatures functions to find all of the features in the located area, as seen in Fig. 2.

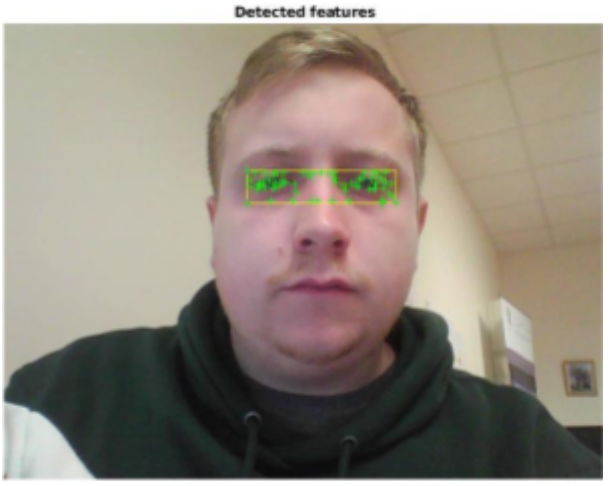

Fig. 2

Each point is stored in a cornerPoint object which contain a number of locations, metrics and a count of them. The location is an x-y of where the feature is located and the metric is a rating of how strong the feature is. I will be using these to create the model.

C. Model Building

To build the model we take the first 10 metrics and their locations and concatenate them to make a list of vectors. This is done as each image can produce any number of metrics. And this will provide a uniform vector with which to use as a predictor. Before training the model the data is put into a random order. And then split into two subsets. The first will be used for training the model and the second will be used for testing it.

The classification technique used for this is a State Vector Machine(Osuna, Freund and Girosit, n.d.). SVM is a supervised learning technique which takes in a list of predictors, and a list of which of two categories the predictors belong to. This suits our needs are we only want to predict if the state of a users eyes are open or closed.

Matlab comes with a function called fitcsvm. This takes in the metrics and value subsets and creates an SVM model that can be used to predict eyes state of future images.

D. Model Testing

Testing the model is done using the testing subset. We use the Matlab predict function with the model and the subset containing the testing predictors and compare the result of each predictor to the corresponding value. This method gives us a good idea of how the model performs.

A second test was also used. Where an image similar to fig. 1 and fig. 2 was used as a “litmus” test to verify the accuracy score.

E. Results

The model performs very poorly. A model created with over 300 images and tested with over 300 more, managed to predict the correct eye state 54.78% of the time. This varied each time the model was created as the training and test data is randomly selected. But at just over 50% it might as well be guessing at random.

F. Conclusion

After a lot of adjustments nothing seemed to improve the model. Initially I was just using the metric strength without limiting the number of metrics allowed to predict, however this still gave poor results. As matlab requires having a standard size for the vector. This means that the vector had to be the size of the maximum vector which was 571. Where most vectors were less than 60, so the rest of the values were filled with 0.

The second adjustment was to only use metrics with a certain number of metrics. To determine this I initially used the 3-Sigma rule(Encyclopediaofmath.org, 2019). This process will give a range 3 standard deviations above and below the mean. Everything outside this range is eliminated. However This still did not give an accurate representation of the most common number of metrics.

The next step was to manually set the min and max range manually. A range of 10 to 30 metrics, at first glance showed signs of improvement in the model. Further narrowing the range yielded more improvements in the accuracy rating. However it could not pass the second test. Upon further investigation the increased accuracy was due to open eye images being discounted as most of them did not fit within the set range.

Classification is not the only means of predicting based on a set of values. Regression is a similar process where the data is plotted and a line of best fit is drawn. Then future data is put on that line to predict the result. However, this also produced an inaccurate model.

Creating a regression model lead to discovering the issue with this method. There is no identifiable differences between the features of the open eyed images and the closed eye images. A number of factors contribute to this. The first is the difference in the data. The images are from two different data sets and are of numerous different people in different positions. The second is also down to the data. The cornerPoint objects store locations and metrics, but there is no way to tell what the metric or location is for, this makes it impossible to compare the data.

For this to work I would have to make my own data set containing a number of images for each person, with their eyes open and then closed in a set range of different positions. This way the data gathered will give more indication of the differences between a person with open eyes and closed eyes. Instead the data used is too sporadic. The lighting, position and facial features are different in each image. Making it hard to extract comparable eye features with which to create a model.

IV. Circle Detection Technique

A. Cropping

When looking at specific features of an image it is a good idea to get rid of the parts you do not need. For this method we will just be looking at the eyes, similar to the classification method. Using the CascadeObjectDetector specified to get the eyes, we get a rectangle which the eyes are in. The function does not get quite enough however. So we get the box from the function then make it a bit bigger so there is a better view of the eyes, as seen in fig. 3.

Fig. 3

B. Edge Filtering

Once we have the cropped section of the eyes we need to prepare it for circle detection. One method of making shapes more apparent in an image is to filter out the edges. This process can be done using a number of different algorithms. Canny and Sobel being the most popular(*Mohan, Vijayarani. (2013)).* After some experimentation I found that Sobel produced a more suitable image for this application. As there was less noise and the eyes are more clearly visible.

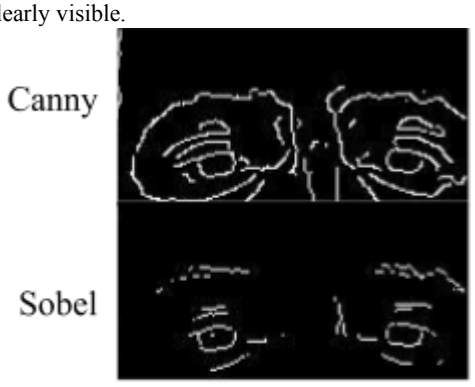

Fig. 4

As you can see in fig. 4, the sobel method has less noise and there are fewer shapes that could be seen as eyes.

C. Circle Detection

After filtering the image and edges we can now see if we can find circles in the image. These circles are the eyes we are trying to detect. The most common method of detecting shapes is known as the Hough Transformation. Matlab has a built in function to detect circles imfindcircles. Imfindcircles uses a Circular Hough Transform based algorithm for finding circles in images. This approach is used because of its robustness in the presence of noise, occlusion and varying illumination. This function also takes in a range of radii for acceptable circles. This is used to further filter out other circles detected in the image.

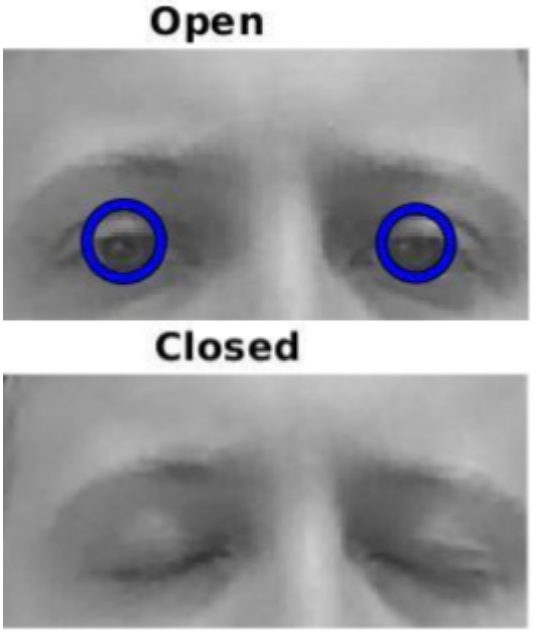

Fig. 5.

D. Results

This method performed very well. Being able to identify if the person in an image had their eyes open or closed the vast majority of the time.

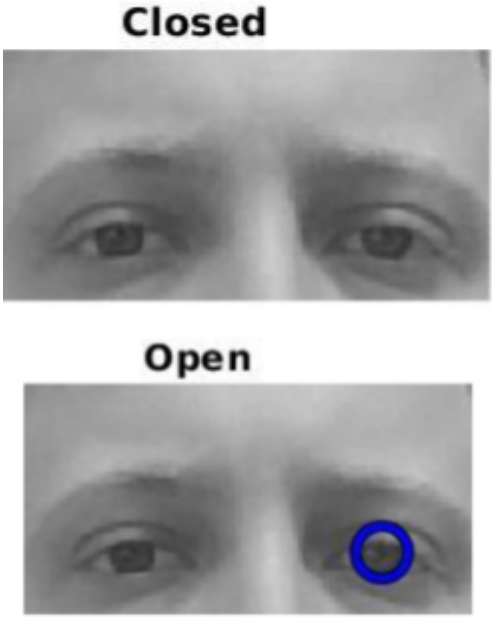

Fig. 6

Fig. 6 contains examples of when there are errors. These come about when it cannot identify circles in the images. However this does not occur often.

E. Conclusion

This method is quite effective for determining the eye state of a person. Given the person is facing the camera. This method would not be effective in situations where the camera is not continuously pointed at the intended target. However due to how controlled the driving seat in a vehicle can be this method can be easily implemented and be effective as part of a driver monitoring system.

V. Implementation

A. Technique Used

Based on the results of the two techniques investigated I will be using the circle detection method. As this method works, and is much simpler than the classification technique.

B. Camera Location

A driver monitoring system will have a camera placed on the dashboard which will look up at the driver, as seen in fig. 7. This means that the camera will have a constant view of the drivers face and there will not be a lot of variations in the position the drivers face.

Fig. 7

C. Logic

When the system detects that the user has their eyes closed it will need to be able to tell the difference between a blink and a when the driver is asleep. To do this the system works on a timer, the timer starts at 0, and if the driver's eyes are closed for 10 consecutive frames (This can be adjusted to match appropriate times) the user will be warned to stay focused. If the drivers eyes remain closed for a further 20 frames (30 frames in total) the vehicle will be pulled over. This is represented in matlab by stopping the program. If the system detects that the user has their eyes open the time is reset to 0.

D. Benchmark Testing

To test the system a video was recorded of myself. I initially had my eyes open, then closed them for a short period of time. This was to initiate the driver warning function. Then after opening my eyes for a while. I closed them for longer, so that the vehicle stop function would be triggered.

E. Results

When the video was played with the driver monitoring system it performed very well. Being able to tell when my eyes were open or closed most of the time. There were a number of errors, the system can have trouble identifying the eyes when the driver has their eyes open. But is good at identifying when the driver's eyes are closed.

The misidentification issues are okay since they are not very common and they mostly misidentify if the eye state when the driver's eyes are open. This error also only occurs for 1-2 frames, which means the misidentification does not trigger the safety features.

The two safety features were triggered as expected. At the 10 frame and 30 frame mark. These would need to be adjusted to match with appropriate times. For example 5 seconds for the warning and 15 seconds to pull the vehicle over.

F. Conclusion

The eye state monitoring would only be a small part of the monitoring system. As shown errors are unavoidable and thus this system should not be solely relied upon for monitoring the weariness of the driver. Other systems such as a heart rate monitor, a thermal imaging camera should be used in conjunction to monitor the state of the driver.

VI. Conclusion

Initially when starting this project the aim was just to use a classification algorithm to build a model which would then be used in the implementation of a DMS. However given the time available this was unachievable due to the reasons stated in III.F. As a result I had to investigate another method of monitoring a drivers eye state. This lead me to discover a simpler and more effective way to monitor eye state. Which performs well and achieved the original goal.

VI. References

Vis-cs.umass.edu. (2019). *LFW Face Database : Main*. [online] Available at: http://vis-www.cs.umass.edu/lfw/ [Accessed 8 Dec. 2019].

Parnec.nuaa.edu.cn. (2019). [online] Available at: http://parnec.nuaa.edu.cn/xtan/data/ClosedEyeDatabases.html [Accessed 8 Dec. 2019].

Yao Hongxun, Gao Wen, Liu Mingbao and Zhao Lizhuang (n.d.). Eigen features technique and its application. *WCC 2000 - ICSP 2000. 2000 5th International Conference on Signal Processing Proceedings. 16th World Computer Congress 2000*

Osuna, E., Freund, R. and Girosit, F. (n.d.). Training support vector machines: an application to face detection. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

Encyclopediaofmath.org. (2019). Three-sigma rule - Encyclopedia of Mathematics. [online] Available at: https://www.encyclopediaofmath.org/index.php/Three-sigma_rule [Accessed 8 Dec. 2019].

Mohan, Vijayarani. (2013). Performance Analysis of Canny and Sobel Edge Detection Algorithms in Image Mining. International Journal of Innovative Research in Computer and Communication Engineering. 1760-1767.