DETECTING SLEEPING DRIVERS

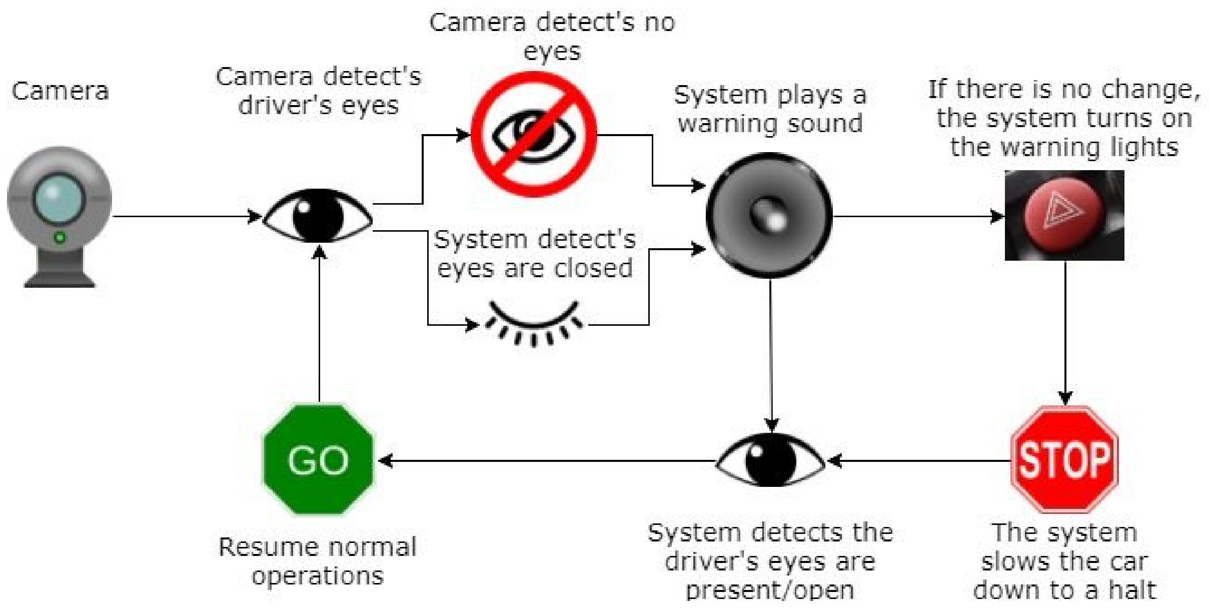
By

Nour Din Saffour, Yann Regev, Ilmari Kaskia

Introduction

Sleep-deprived driving is a common phenomenon, that is a factor in more than 100.000 car crashes, resulting in approximately 6500 deaths and 80.000 injuries annually. Our project is to create a system that detects if the driver of a car is falling asleep behind the wheel through eye detection, and warn the driver with a sound, and if that fails, turn the warning lights on and slow the car down. The aim of this system is to reduce the risk of accidents on the road caused by drowsy and inattentive drivers.

Block diagram:


Camera
Camera detect's driver's eyes
Camera detect's no eyes
System detect's eyes are closed
System plays a warning sound
If there is no change, the system turns on the warning lights
STOP
The system slows the car down to a halt
System detects the driver's eyes are present/open
GO
Resume normal operations

System design

First steps:

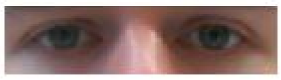
Our first step was to read the literature so we can learn from the experiences of those who have done something like this before. After that, we began designing our system in earnest.

Recognizing the eyes:

The first step was to recognise the eyes of the user. For that, we needed to detect the face of the user in an image and then extract a picture of the eyes from that image. We found a Matlab code to find the eyes in a picture and draw a rectangle around them. From that, it was easy to use the dimensions of the rectangle to crop the picture to include only the eyes:

```
EyeDetect = vision.CascadeObjectDetector('EyePairBig');
BB=step(EyeDetect,im);
im2 = imcrop(im,BB);
```

A cropped image with the eyes only from the program



Detecting the eyes in different lighting situations:

Taking into consideration the different lighting in every environment, We tried to overcome this problem by calibrating when the program starts.
At the beginning of the program the user will be asked to keep their eyes open while we detect the right threshold which will provide us with only two BLOBs.

After we have decided the threshold we will calculate the formfactor of the eyes in the open state. We will use this form factor to decide whether the eyes are open or closed.

Matlab Code:

```
video=webcam;          %start the video camera
display('open your eyes');      %ask the user to open their eyes
pause(1);                       %wait until they react
threshold = 0.0;                %initiate the threshold variable
im=snapshot(video);             %take a photo
EyeDetect = vision.CascadeObjectDetector('EyePairBig');   %detect the eyes
BB=step(EyeDetect,im);
try
        im = imcrop(im,BB);    %crop the eyes image
        im = rgb2gray(im);     %turn it into grey image
catch ...
        display('failed code 1');          %if the eyes weren’t detected display it
        return
end

numlabels = 0;           %initiate the number of labers
%if the number of labels is smaller than 2 keep trying to detect to get to 2 labels
while numlabels < 2
        im2 = im2bw(im, threshold);
        im3 = imcomplement(im2);
        im4 = imclose(im3,strel('disk',6));
        [labels,numlabels] = bwlabel(im4);
        threshold = threshold + 0.005;
end
threshold = threshold + 0.04; % this number was tested to be the best
display('open your eyes');
pause(1);


%take a photo with the new threshold and calculate the form factor to be used in the
program
im=snapshot(video);
EyeDetect = vision.CascadeObjectDetector('EyePairBig');
BB=step(EyeDetect,im);
try
        im = imcrop(im,BB);
        im = rgb2gray(im);
catch ...
        display('failed code2');
        return
end
im2 = im2bw(im, threshold);
```

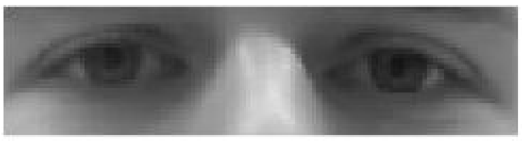
```
im3 = imcomplement(im2);
im4 = imclose(im3,strel('disk',6));
[labels,numlabels] = bwlabel(im4);
stats = regionprops(labels, 'all');
fOpen1 = 4*pi*stats(1).Area/((stats(1).Perimeter)^2);
fOpen2 = 4*pi*stats(2).Area/((stats(2).Perimeter)^2);
```

Detecting the state of the eyes:

Now that we have the picture of the eyes we needed to detect the state of the eyes. We used the skills we learned from the labs. We obtained a usable picture via the following steps:

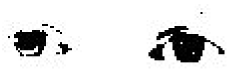
- **Converted the image to a grey image.**

A grey image with the eyes only from the program



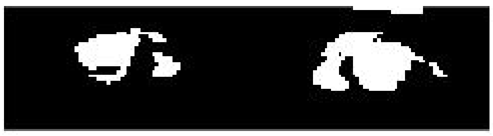
- **Measured the threshold we needed to get only two BLOBs. After that we converted the image to a black and white image using the threshold we measured.**

A black and white image from the program



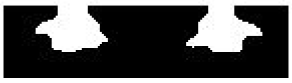
- **Complemented the image to flip the colors**

The black and white image after complementing the colors

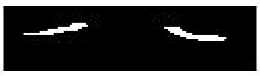


- Applied the morphological closing filter using a disk shape.

A final phase photo from the program with the eyes open



A final phase photo from the program with the eyes closed



Now that we have everything set up we will measure the form factor of the two shapes and depending on the values we will get the program will decide whether the eyes are open or closed.

Deciding the state of the driver:

The fact that the eyes are closed or open in one image doesn’t mean the driver is sleeping or awake. The driver might blink or turn his head left and right which will reflect on the detected state. We decided to judge the state of the driver using a ratio of the open and closed images according to the following parameters:

Samples = 5
Ratio = Eyes open frames / Eyes closed frames
If the ratio is bigger than or equal to 2, the driver is awake.
if it's lower, the driver is sleeping

Matlab code:

```
if (counter >= 5)
        counter = 0;                      %reset the counter
        result = (eyesOpen/eyesClosed); %calculate the ratio
        eyesOpen = 1.0;                   %reset the variables
        eyesClosed = 1.0;
        if (result <= 2.0 )               %decide the state of the eyes
        display ('Sleeping'');
        sound(y,fs);                      %sound the alarm
        speed = 0;                %stop the car
        else
        display('awake'');
        speed = 30;               %keep on driving
        end
```

Controlling the car:

Due to the scope of the project, we did not have access to an actual car, so we simulated our system using a Lego Mindstorms NXT robot. The control specifics of the systems were a big question from the start. Should the car come to a complete stop, how long should the delay between the eyes closing and the system reacting? At what rate should the car slow down? Should the car pull over, or keep it's lane? In a suburban or urban area the car needs to slow down fast since there could be pedestrians nearby, while on a highway stopping too fast would pose a danger to the other drivers. In the end we decided to simulate an urban setting, where the car needs to stop quickly if the driver does not maintain control of the car.

Risk analysis

- Detecting eyes is easy through matlab, but reliably detecting if eyes are open or closed is much more difficult. Our primary aim is to get it to working at an acceptable level for all eyes.
- Eye detection can be compromised if the user is wearing glasses, or shades. Possible problems with deformed eyes or only one eye.
- Different skin colours and lighting conditions might also affect the eye detection

To mitigate these risks, our first aim is to make the eye detection work on an acceptable level before moving on to other parts of the system. The largest part of our project is that we can detect if the user's eyes are open or closed.

Testing

After we had the code in a state we deemed ready, we started testing in earnest. We decided to test it by each of us running the system for approximately 60 seconds, or 60 "frames", and counting how many misses we had in the test period. A miss constituted as the system either detecting no eyes, or detecting the eyes open when they were closed, or vice versa. We received the following results:

Yann round 1: 1 miss/60 frames
Nour round 1: 2 miss/60 frames
Ilmari round 1:3 miss/60 frames

For round 1, we had very few misses within one minute, the whole system having a <5% miss rate.

Yann round 2:5 miss/60 frames
Nour round 2: 11 miss/60 frames
Nour round 2 retry:1 miss/60 frames
Ilmari round 2: 6 miss/60 frames

For round 2, we had a bit more misses, but still generally remained within <10% miss rate. For Nour round 2, we got to almost a 20% miss rate, which we deemed unacceptable; we deduced that this was because of bad calibration. We tried again, and got it down to only 1 miss.

We deemed that a miss rate of less than 10% is acceptable for our system, since the rate of pictures taken would be very fast, and if only 1 picture in 10 would be a miss, it would not pose a serious issue regarding correct performance of the system.

Ethics

First and foremost, we are creating this system with road safety in mind. Driving when tired, or being inattentive behind the wheel are real problems that we want to mitigate. However, this system, if implemented in real life situations, is not a catch-all solution to address this problem. The causes behind it (long working hours/unconventional working hours, not enough rest for drivers, very long trips) need to be addressed properly to fix the problem. The idea behind this project is not to provide a solution to all the underlying problems, only mitigate the effects.

With cameras and filming people, there is always a question of privacy. We seek to get around this problem by having a closed system with no outside connection, and by not saving any data of the user - the system must only sense the immediate situation and react to it. It doesn't need to remember what has happened before, or anticipate what is going to happen.

Lastly, there is the question of giving a machine control over your car with people inside. This is not a completely self-driving car with it's myriad of ethical issues, but it does skirt the surface of this issue. For instance, if the system malfunctions and accidentally stops the car even when the driver is attentive and awake, and thus causes an accident, or even just leaves the driver stranded, is it considered an unfortunate circumstance, or will the creators of the system be held responsible? Or, if the system does not work, the driver falls asleep and causes an accident, is it the system's fault, or the driver's, as it would have been if the system was not installed? Is the system considered reliable enough to be held responsible for the accident, or is it only a mitigating factor, with the blame landing on the driver himself?

These are issues that would need to be resolved before the system could be released for commercial use.