International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 03 | Mar-2018 www.irjet.net p-ISSN: 2395-0072

Drive Assistance an Android Application for drowsiness detection

Priyank Thakur[1], Mithilesh R.Sarmalkar[2], Rahul A. Gupta[3], Sanket R. Kamble[4], Prof. Pravin Hole[5]

[1 2 3 4] Department of Computer Engineering, Terna Engineering College, Nerul(W)
[5]Professor,Department of Computer Engineering, Terna Engineering College, Nerul(W)

Abstract: *With ever rising number of cars on road has lead to an increase in the accidents on the road. The major factor responsible for road accidents is human error. This paper presents an application which is intended to provide assistance for drivers who feel drowsiness while driving and alert them. This Android application will use its processing power to continuously monitor driver and provide the driver with real-time data regarding the car's surrounding and show this information on the android application. For face and eye detection Viola-Jones Algorithm [1] is used. For face and eye detection a pre-trained classifier is used. And on the basis of template matching a sound or alarm is raised for notifying driver regarding drowsiness*.

Keywords: Android, Viola-Jones Object detector, cascade classifiers, OpenCV, Arduino.

1. Introduction

According to the report compiled by the Transport Research Wing, the total number of road accidents increased by 2.5 per cent from 4,89,400 in 2014 to 5,01,423 in 2015 [2]. The total number of persons killed in road accidents increased by 4.6 per cent from 1,39,671 in 2014 to 1,46,133 in 2015.

The major reason contributing to these statistics is human error or drowsiness.

In this system, computer vision for detection of face and eye blink and based on eye blink warns driver if he is feeling drowsiness.

This system uses real time video from secondary camera and performs image processing on the frame captured from the video and based on if eye are closed raises and alarm alerting thus alerting driver. The main goal of the system is to use the processing power of android application to perform computer vision related tasks thus making the system across a wide number of devices running Android OS. Arduino is also used to improve further drivers awareness by displaying if any car comes near the car or in dead spot region of car's surrounding.

2. Drawbacks of current system

The current systems which are used require the use of dedicated hardware for performing function of eye detection and hardware at times can be expensive.

3. Proposed methodology and implementation:

The System contains following components

1. Android application
2. Image processing
3. Arduino interfacing to android phone for providing sensor information

Android application is used as a core component of a system. It will display information like speed of the car, navigation details and recent activity. To present the previously mentioned data in a proper manner, android fragments are used [3]. This not only improves the aesthetics of the User Interface but also helps to group related data together.

To add extra functionality to the system sensors like ultrasound senor will be placed around the car which will detect for objects around the car and pass the collected information to android application for processing. Sensors placement around the car will be in such a manner around car where drivers tend to have a blind spot and are usually difficult for driver to look as they have to take their eye sight of the road to look to it. This data from sensor is passed to android application through use Arduino to which the ultrasound sensors are connected. This data collected then can be showed on the android application with the help of graphics.

For detection of the face and eye region, object detection proposed by Paul Viola and Michael Jones is used [1]. In their paper Viola and Jones proposed an object detector through the use of Haar feature extraction. For Haar feature extraction a set of cascades is used. These cascades are combination of simple classifiers which combined gives a quite robust cascade classifier for eye extraction. Based on object detection, application will take the decision to raise alarm or not. For image processing we are using OpenCV libraries with C++ code for object detection. OpenCV libraries are used in this application as they have native support for android development and quite a large no of optimized algorithm and methods for performing operations on images or videos [4].

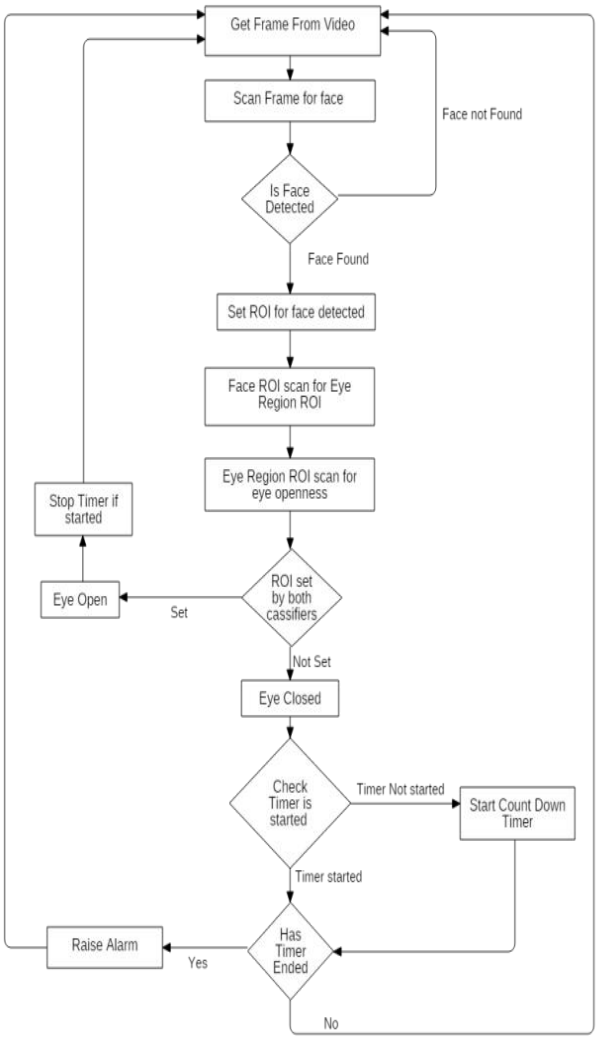

Fig. 1 Flowchart for alarm Raising based on eye detection

Figure 1 shows us flowchart used for eye blink detection in application. First we obtain frame from video which is captured through secondary camera of android phone. A call to face and eye region detection method is made through Java Native Interface (JNI). First we load trained classifiers into a cascade classifier for object detection. After loading the cascades into classifier, the frame is scanned for faces and Region-of-Interest (ROI) is set around face detected. This ROI is then again scanned for Eye Region and Eye region ROI is set. Eye Region ROI is then again scanned for Eye Open or closed detection. If both cascade detect eyes then eyes are open if only one ROI is detected for eye then eyes are closed.

Three cascades are used for working of the application. The three cascades used are as follows:

1. haarcascade_frontalface_alt
2. haarcascade_eye_tree_eyeglasses
3. haarcascade_lefteye_2splits

The above mentioned cascades are provided along with OpenCV libraries. These cascades are trained by 6665 positive libraries from FERET, VALID and BioID face databases [6]. Haarcascade_frontalface_alt is used for detecting faces from the image frame, haarcascade_eye_tree_eyeglasses is used for detecting eye region from face region. This cascade is capable of detecting eyes even if the image contains person wearing glasses. Haarcascade_lefteye_2splits is capable of detecting both eyes open and closed.

These cascades are loaded into cascade classifier class of OpenCV [5] using load function. To perform object detection detect Multi Scale function is used. In detect Multi Scale function Mat object of input frame is passed along with vector in which we need to store ROI pixels and scale Factor parameter which specifies how much the image size is reduced at image scale and maxSize parameter which defines maximum possible object size.

Whenever input frame is captured it is first converted into gray scale as OpenCV object detection functions work only on gray scale images. Input frame is stored in Mat object and to improve speed of object detection input frame is scaled down to image resolution of 800x600 pixels with maintaining proper aspect ratio in order to avoid distortion of image. After detectMultiScale is used, results are stored in three different vectors namely face vector, eye vector and eye_open vector. In order to detect if the object is detected or not it can checked by simply checking size of each vector. Further to improve object detection the input frame image is divided into two parts Left face ROI and Right face ROI.

Also for proper eye detection to take place first face detection needs to occurs and then eye detection needs to take place. The classifiers function in following sequence.

Fig. 2 Classifier Sequence of working

Based on the output of above classifiers if eye closed for a certain time period an alarm will be raised. In order to avoid stopping of object detection function a separate thread will be fired which will raise the alarm. This thread will be run on application UI thread for better optimization and this thread will use Media Player class of android for playing sounds [3]. For thread to run on UI thread, it must run in android application as native thread is managed by Linux Kernel in android.

4. Experiments and Results

Face detection and Eye detection work nearly at speed of nearly 10-12fps. Due to using pre-trained cascades provided with OpenCV libraries, object detection can get confused and

detect and track wrong objects like nostrils. However this problem can be solved by training a better cascade with more positive and negative sample images then previously used datasets. Fig 3 shows eye detector when eye are open and Fig 4 shows eye detector when eyes are closed.

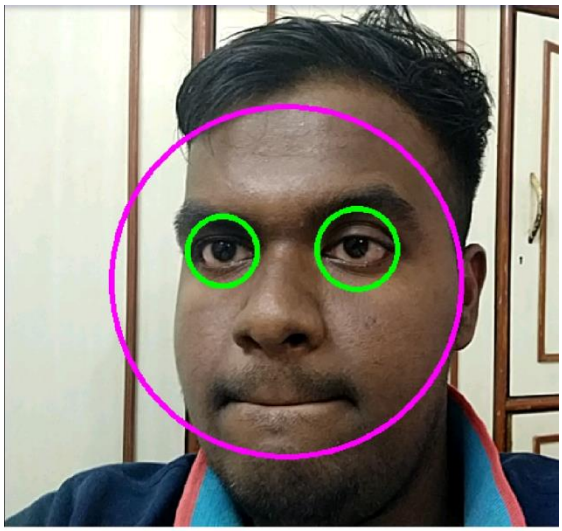

Fig. 3 When Eyes are open.

Fig. 4 when eyes are closed

5. Experiments and Results

So we can conclude that drive assistance has potential to be used in future markets. Due to the project being based on mobile platform it has quite the potential to be widespread. Also the concept of IOT can be brought too many cars thus being able for cars to communicate with one another. Also this project can help reduce the accidents caused by the negligence of driver and accidents caused due to drowsiness. Further enhancements such as history of routes taken, drive details can be added in future to further improve functionality of the project.

REFERENCES

[1] Paul Viola, Michael Jones "Rapid Object Detection using a Boosted Cascade of Simple Feature" ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001.

[2] ROAD ACCIDENTS IN INDIA 2015 REPORT, GOVERNMENT OF INDIA MINISTRY OF ROAD TRANSPORT & HIGHWAYS TRANSPORT RESEARCH WING

[3] Android Developer tutorial: https://developer.android.com

[4] OpenCV Libraries: https://docs.opencv.org/2.4/modules/refman.html

[5] OpenCV CascadeClassifiers: https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html

[6] Feret Database: https://www.nist.gov/itl/iad/image-group/color-feret-database