International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 6, ISSN No. 2455-2143, Pages 106-110
Published Online April - May 2016 in IJEAST (http://www.ijeast.com)

DROWSINESS DETECTION SYSTEM FOR DRIVERS USING HAARTRAINING AND TEMPLATE MATCHING

Lorraine Saju, Christeena Jestine, Farhana Yasmin, Surekha Mariam Varghese
Computer Science and Engineering Department,
Mar Athanasius College of Engineering, Kothamangalam, Kerala, India

***Abstract*—Driver impairment due to drowsiness is known to be a major contributing factor in many motor vehicle crashes. More than 30% of the road accidents are caused by the fatigue of the driver. At present, there are various drowsiness detection systems available in the market. These systems are implemented using any one of the various implementation techniques such as detection of any behavioral pattern, changes in physiological conditions, or vehicular motion. Consequently, the accuracy of such systems has been found to be low. Here, we combine various changes in the facial expression in order to achieve more accurate results. The purpose of this system is to reliably quantify commercial motor vehicle driver drowsiness and provide a real-time warning to the driver. The measures that can be used for the purpose are eye closures, frequency of yawning, head tilt, etc. The system has been implemented by means of a camera, a Raspberry pi board and corresponding alert system in the form of a buzzer. Around 7000 facial patterns were used to train the system to obtain accurate results.**

***Index Terms* — Cascade Classifier, drowsiness detection, haartraining, OpenCV, Raspberry pi, template matching**

I. INTRODUCTION

A major share of road accidents that happen today are caused by the drowsiness of the driver. Many lives could be saved if there was a mechanism by which the drivers could be given an alert when they start feeling drowsy. Statistics indicate the need of a reliable driver drowsiness detection system which could alert the driver before a mishap happens. This paper aims at developing such a system that can observe the mannerisms of the driver, obtain relevant data, process the information and produce the alert according to the situation at hand. Research has shown that there are basically three ways to determine drowsiness in a driver: vehicle-based measures, behavioural patterns, and physiological measures [1], [2].

This paper uses the behavioural measures of the driver to determine whether the driver is drowsy or not. We try to combine two of the behavioural measures that a driver can display when he/she feels sleepy while at the wheel, vis-à-vis: movement of the eye, and movement of the head and facial changes. When sleepy, every person has a tendency to blink frequently or to shut their eyelids in slow movements. Another behavioural measure displayed by human beings when drowsy is frequent yawning. Also, for most people, the manner in which their head is kept changes abruptly as they fall asleep. That is, our head tends to swing down a few angles in an instant when feeling drowsy. We have developed the software tools to detect and track the aforementioned behavioural patterns, record the data, process the information and produce the desired result according to the output produced, such as, an alert to wake the driver up before the scene goes awry.

Most of the published studies on using behavioral approaches to determine drowsiness focus on blinking. This measurement has been found to be a reliable measure to predict drowsiness and has been used in several commercial products. However, it has been known to produce inaccurate results at times. This paper puts forward a system that attempts to combine multiple behavioural attributes such as eyelid movement, head alignment, yawns, etc. to detect drowsiness in the driver since obtaining data from multiple inputs have more chances of improving the accuracy of the system.

The system has been implemented using the input from a camera that is to be fixed in the car, in front of the driver. The camera records eyelid movements, lip movement, and the head movement after which this data is processed producing the corresponding output.

Whenever the user closes his/her eyelids for more than a predefined period of time, the alert goes off. This is detected by comparing and matching the images used in the training phase of the system with the real time images captured by the camera. The comparison and matching is done after each

International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 6, ISSN No. 2455-2143, Pages 106-110
Published Online April - May 2016 in IJEAST (http://www.ijeast.com)

frame, and the user is given an alert upon detection of closed eyelids in a certain number of continuous frames.
The next sections of the paper discuss technical details of the project such as the hardware used, various technologies used to implement the system, etc. It then goes on to explain the implementation details of the prototype system which has been executed by means of the Raspberry pi single-board computer.

II. BACKGROUND

2.1 Ensemble Learning

Ensemble learning is a machine learning process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem. Ensemble learning is primarily used when improvement in the performance of the system is needed, or to minimize the probability of producing poor output. The whole process involves classification, prediction, function approximation, etc. Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, non-stationary learning and error-correcting. This paper focuses on cascade classifier of ensemble learning, which consists of various stages, each stage of which is a collection of weak learners.

2.1.1 Cascade Classifier

The cascade classifier is a part of ensemble learning, which is implemented in various stages that consists of simpler classifiers. These are applied to a region of interest until input is accepted or rejected.

First, a classifier (namely a *cascade of boosted classifiers working with haar-like features*) is trained using a few hundred sample views of a particular object (i.e., a face or eyes in this case), called positive samples, and negative samples, which are arbitrary images. Every positive and negative sample is scaled to the same size.

After a classifier is trained, it can be applied to a region of interest in an input image. This region should be of the same size as the samples used during the training. The output of the classifier will be "1" if the region is likely to show the object that we are looking for (i.e., face/eyes), and "0" otherwise. It is also possible to search for the object in the whole image by moving the search window over the image and checking every location using the classifier. The classifier is designed so that it can be easily "resized" in order to be able to find the objects of interest at different sizes. This procedure is found to be more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales.

2.2 OpenCV

OpenCV (*Open Source Computer Vision*) is a cross-platform library of programming functions mainly aimed at real-time computer vision. It is free to use, and provides an array of functions for machine learning, neural networks, and other applications of artificial intelligence. To support machine learning, OpenCV provides a statistical machine learning library that contains boosting, decision tree learning, Artificial Neural Networks (ANN), Support Vector Machine (SVM), ensemble learning, etc.

It is written in C++ and the primary interface is in C++ as well. There are bindings in Java, Python, and MATLAB/OCTAVE. To encourage adoption by a wider audience, wrappers in other languages such as C#, Ch, Perl, and Ruby have been developed. All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

2.2.1 HaarTraining

The OpenCV library provides a great number of functions for face detection, feature detection such as eyes, mouth, sunglasses, etc. Furthermore, it provides programs (or functions) that are used to train classifiers for their face detection system, called HaarTraining [4]. We can create our own object classifiers using these functions.
Object Detection using Haar feature-based cascade classifiers is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.
The algorithm extracts images using a lot of positive (faces, eyes, etc.) and negative images (arbitrary images without faces). Each feature is a single value obtained by subtracting sum of pixels under various regions of the images. Features include edge features, line features, etc. For each feature, the pixels used for extraction differs.
Now all possible sizes and locations of each image kernel are used to calculate plenty of features. But all the features extracted will not be useful for the purpose at hand. For example, in this paper only eyes, mouth, and the face are relevant. Other facial features like cheeks, nose, forehead, etc. are useless as far this paper is concerned. To extract only those features relevant to the project, we use a technique called Adaboost.
For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Then we select the features that display the least error rate (best threshold). Initially each feature is given an equal weight. As the process continues, the weights are updated according to the results obtained so that accuracy is improved. The process is repeated until the required level of accuracy or the required number of features is found.
Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but

International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 6, ISSN No. 2455-2143, Pages 106-110
Published Online April - May 2016 in IJEAST (http://www.ijeast.com)

together with others forms a strong classifier. With around 6000 images, even 200 features can provide 95% accuracy [3]. In an image, most of the image region is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Don't process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.
For this they introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it. We don't consider remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region. This is how face detection and other feature detection is done using haartraining by means of cascade classifiers.

2.2.2 Template matching

Template Matching is a method used for finding whether a template image is present in a larger image or not. OpenCV comes with a function matchTemplate() for this purpose. To do that, the function simply slides the template image over the larger image and compares the template and patch of input image under the template image. There are several comparison methods available in OpenCV.
The function returns a gray scale image as the output, which will tell you how much of the template image matches with the input image. Once you got the result, you can use minMaxLoc**()** function to find where the maximum/ minimum value is.

2.3 Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It is capable of everything that a normal desktop computer can do, from word-processing, and playing games, browsing the internet and playing high-definition video, to making spreadsheets, etc.

The Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. It can is an effective tool to perform functions like digital signal processing, and image processing.

III. SYSTEM DESIGN

The system is designed as an independent unit, with a camera, a processor, and an alert system. The camera should support high definition capture for optimum performance. Since the system has to be flexible enough to use any time of the day, lack of light must never be an issue. Therefore, a nonintrusive light, like the IR can be provided. This would ensure ample lighting independent of natural light that will not disturb the driver, but will be detected by the camera. The video thus captured, is given as the input to the processor. The processor must be capable of image processing, with at least 1 GB of memory, x MHz speed and 2 GB RAM. The processor is connected to a buzzer which acts as an alert system. When required, the buzzer beeps, alerting the user to focus on the road or take a break to refresh.

1. Sample Collection

The system is almost entirely dependent on face detection, and hence the primary step towards the development involves the creation of a system to detect human face and its basic features. Face detection is done using Cascade Classifier, which requires a very large set of samples for training. Almost 7000 images each of the face, eyes, and mouth were collected from various sources. An equal amount of negative images were also collected. While the positive image shows an object that must be detected, the negative image shows a similar picture with the object missing from it. For extracting the target object from the positive image, a small program called 'object_marker.exe' has been used.

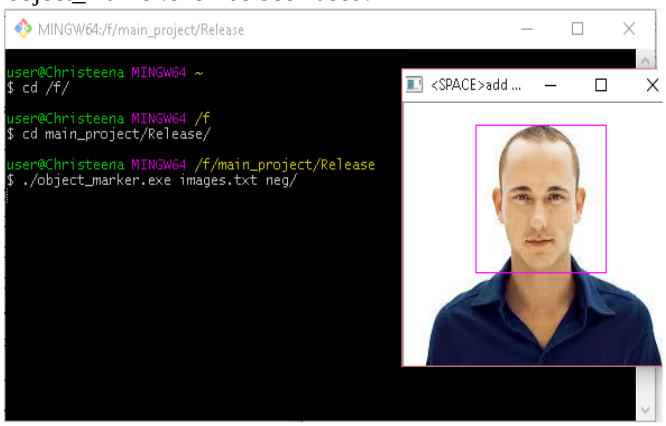

Fig 1. Extraction of features

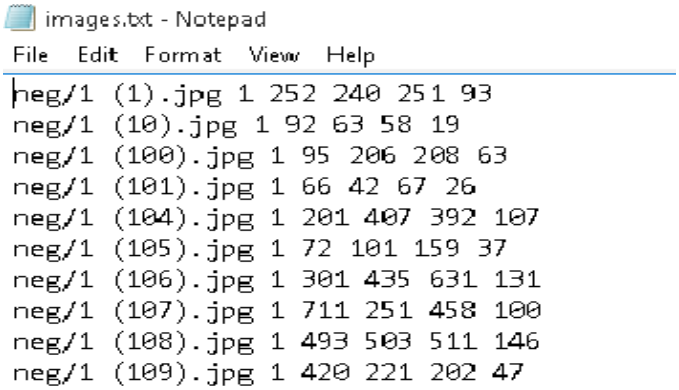

Fig 2. Extracted images stored as text file

The target objects thus extracted (figure 1) from the positive

International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 6, ISSN No. 2455-2143, Pages 106-110
Published Online April - May 2016 in IJEAST (http://www.ijeast.com)

samples will be stored as a text file. Figure 2 shows a snippet of the text file. The first field indicates the name of the image. The second field indicates the number of target objects detected, the third and fourth fields give the x and y coordinates, respectively of the top left corner of the rectangle denoting target object's location.

2. Sample Creation

OpenCV library provides a number of programs that can be used to train classifiers for face detection system called haartraining. The utility opencv_createsamples is used to prepare a training dataset of positive and test samples. It provides functionality for dataset generating, writing and viewing. The large set of positive samples is created from the object image by random rotation and other transformations. The amount and range of randomness can be controlled by command line arguments of opencv_createsamples utility. Fig. 7 shows command used.

The output is a file with *.vec extension, which is a binary format of images.

3. Cascade Training

Here, opencv_haartraining has been used to train a cascade classifier. This process may take a number of days depending on the number of samples and stages of the cascade classifier. Once the application has finished execution, an XML file will be generated in the specified folder. The XML file is the final trained cascade classifier.

4. Implementation

There are two basic algorithms that are explained below which explains how the system works. The first one is the Match Algorithm, which is for template matching and the second is the Drowsiness Detection Algorithm, which forms the core of this paper.

Algorithm 1. Match algorithm
Input: Source image, I
Input: Template image, T
Output: minimum value of the matrix that is created as the result of the matching

1. Match I with T using OpenCV matchTemplate function.
2. Normalize the resulting matrix
3. Find the global minimum in the array, minVal
4. Return minVal

Algorithm 2. Drowsiness detection algorithm

1. Get the input from the camera and store the frame in a matrix, source
2. Detect face, eyes and mouth using the trained cascade classifier
3. Store the images of eyes, face and mouth in normal condition in face_template, eyes_template and mouth_template
4. Repeat while the system is in ON state
   4.1. capture a frame from the camera into the matrix, source
   4.2. face_min = Match(source,face_template)
   eye_min = Match(source,eyes_template)
   mouth_min = Match(source,mouth_template)
   4.3. **if** (count<5) **then**
   avg_face_min = avg_face_min + face_min/5
   avg_eye_min = avg_eye_min + eye_min/5
   avg_mouth_min = avg_mouth_min + mouth_min/5
   count++
   endif
   else
   if (face_min-avg_face_min > CONST1)
   sound alarm
   if (mouth_min-avg_face_min > CONST2)
   detected yawn
   if (yawn detected more than 5 times in a minute)
   sound alarm
   if (eye_min-avg_face_min > CONST3)
   detected eye closure
   if (eye closure detected in 10 or more continuous frames)
   sound alarm

Algorithm 1 matches a template image with a source image with the help of the OpenCV matchTemplate function, which has been explained in the previous section.

Since we use the CV_TM_SQDIFF_NORMED match method, the best matches are lower values. Hence we find the global minimum value of the normalized resulting matrix and return it.

Algorithm 2 describes the working of the system. First, as described in the algorithm, we detect the face, eyes and mouth of the person (in his alert state), looking at the camera. This is done by including the XML files that are the final trained cascade classifiers. We have used three classifiers, one each for face, eyes and mouth.

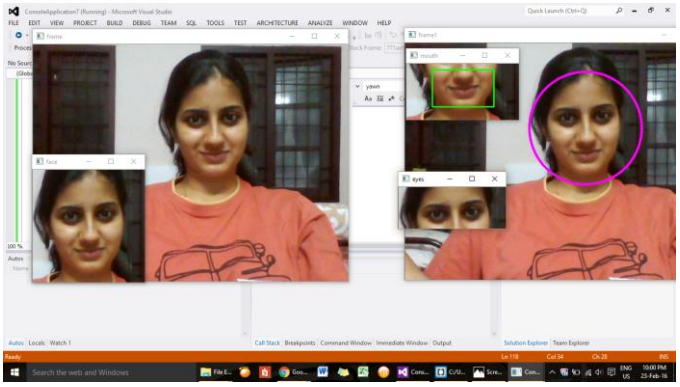

Fig 3. Detection of face, eyes, and mouth

Once detected, we capture them (figure 3) and store them as

International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 6, ISSN No. 2455-2143, Pages 106-110
Published Online April - May 2016 in IJEAST (http://www.ijeast.com)

templates for later comparisons. We store them in matrices. It is these comparisons that help us detect whether the person has become drowsy or not.

Now the system can start functioning. Each frame that is captured by the camera is searched for the three template images that has been previously stored (Steps 4.1 and 4.2), using the Match function explained in Algorithm 1.

For the first 5 frames, we find the average of the return values face_min, mouth_min and eye_min (Step 4.3). For the remaining frames we compare the return values with the average value calculated for face, eyes and mouth. If the difference calculated is more than a constant value (CONST1, CONST2 and CONST3 for face, mouth and eye respectively), then we can assume that a change in the features has occurred. If the change is for face, then that means his face is no longer straight and the alarm is sounded instantly. If the change is for eyes, then it could mean that he is blinking continuously or that he is dozing off. So we make sure the eye is closed for a reasonable number of frames before sounding the alarm. If the change is for mouth, it means he is yawning wide. So if a yawn is detected (figure 4) more than five times in a minute, the alarm will be sounded to alert the driver.

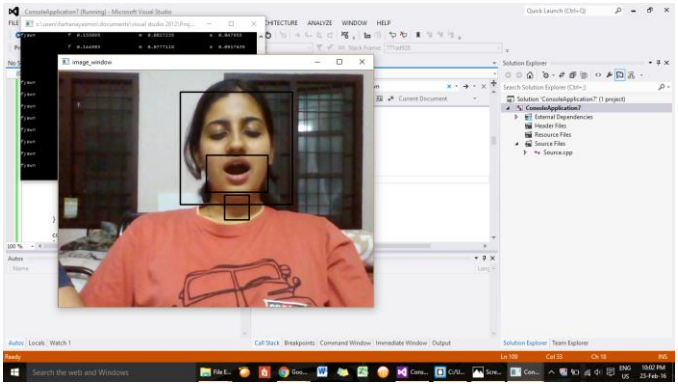

Fig 4. Detection of yawn

The prototype has been implemented using Raspberry pi board. The input from the camera is connected to Raspberry pi, which runs these algorithms on the input to detect drowsiness in the user. The system can be made more powerful and fast by using processing boards that possess higher processing capacities.

IV. CONCLUSION

This project aims at developing a software tool for timely and accurate detection of drowsiness in a driver while at the wheel by considering multiple facial features as inputs. By capturing normal image of the user and comparing it with the input feed, the system detects yawns and eye closures to check whether the user is drowsy or not. Input is captured by the cameras, processed by Raspberry pi, and the output is in the form of a buzzer that gives the user an alert as and when drowsiness is detected. The limitation that can act as a detriment to the precision of this system would be poor lighting. This can be overcome by using an appropriate lighting system, if required.

V. REFERENCES

[1] S. Abdul-Kareem, Haitham Hasan. Static hand gesture recognition using neural networks. Springer Science+Business Media B.V. 2012

[2] Drowsy Driver Detection System, Neeta Parmar, Peter Hiscocks, Department of Electrical and Computer Engineering, Ryerson University.

[3] Paul Viola, Michael Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. Accepted conference on computer vision and pattern recognition; 2001.

[4] OpenCV 2.4.12.0 documentation. Cascade Classifier Training, http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade/; 2015, [Accessed 12-12-2015]