ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

DROWSY DRIVER DETECTION SYSTEM

Mrs. Smita Jawale, Ms. Pragati Malvadia, Ms. Ashwini Meena
Department of Computer Engineering, Vidyavardhini's College of Engineering and Technology, India

DOI: 10.5281/zenodo.48987

ABSTRACT

A Drowsy Driver Detection System is an Image processing based system. This system is developed using a non-intrusive machine vision based concepts. In this system, there is a camera that will be continuously monitoring the driver's face to detect fatigue. In case the driver is detected as fatigue, the system issues an alarm. This system detects drowsiness by checking the amount of time the eyes are closed. The first five consecutive frames of the camera is checked, if the eyes are found closed in all the five frames, then the system issues alarm. Before determining whether eye is open or closed, it is required to determine eye location. To determine eye location, the images captured from the camera are binarized. This binary version of the image help to find the edges of the face, which narrows the area of where the eyes may exist. Once the face area is found, the eyes are found by computing the horizontal averages in the area. Based on intensity change eye location is found, as eye regions in the face present great intensity changes. Once the eyes are located, measuring the distances between the intensity changes in the eye area it is determined whether the eyes are open or closed. A large distance corresponds to eye closure.

KEYWORDS- Computer Vision, Drowsy Driver, OpenCV.

INTRODUCTION

Driver fatigue is a significant factor in a large number of vehicle accidents. A driver who falls asleep at the wheel loses control of the vehicle, an action which often results in a crash with either another vehicle or stationary objects. In order to prevent these devastating accidents, the state of drowsiness of the driver should be monitored. Driver drowsiness detection is a car safety technology which prevents accidents when the driver is getting drowsy. Driver inattention might be the result of a lack of alertness when driving due to driver drowsiness and distraction. The aim of this project is to develop a prototype drowsiness detection system. The focus of the system is mainly on the state of eye. Based on whether the eye is closed or open in all the consecutive five or six frames. If the driver's eyes are found closed in all the frames it detects that driver is fatigue so that the driver can be made alert before it ends up with an accident. The analysis of face images is a popular research area with applications such as face recognition, virtual tools, and human identification security systems. This project is focused on the localization of the eyes, which involves looking at the entire image of the face, and determining the position of the eyes by a self-developed image-processing algorithm.

MATERIAL AND METHODS

What Is Computer Vision?

Computer vision is the transforming of data from a still, or video camera into either a representation or a new decision. All such transformations are performed to achieve a particular goal. A computer obtains a grid of numbers from a camera or from the disk, and that's that. Usually, there is no built in pattern recognition or automatic control of focus and aperture, no cross-associations with years of experience. For the most part, vision systems are still fairly naïve.

The Origin of OpenCV

OpenCV came out of an Intel Research initiative meant to advance CPU-intensive applications. Toward this end, Intel launched various projects that included real-time ray tracing and also 3D display walls. One of the programmers working for Intel at the time was visiting universities. He noticed that a few top university groups, like the MIT Media Lab, used to have well-developed as well as internally open computer vision infrastructures—code that was passed from one student to another and which gave each subsequent student a valuable foundation while developing his own vision application. Instead of having to reinvent the basic functions from beginning, a new student may start by adding to that which came before.

OpenCV Structure and Content

OpenCV can be broadly structured into five primary components. The CV component contains mainly the basic image processing and higher-level computer vision algorithms; MLL the machine learning library includes many statistical classifiers as well as clustering tools. HighGUI component contains I/O routines with functions for storing, loading video & images, while CXCore contains all the basic data structures and content.

Why OpenCV?

Specific

OpenCV was designed for image processing. Every function and data structure has been designed with an Image Processing application in mind. Meanwhile, Matlab, is quite generic. You can get almost everything in the world by means of toolboxes. It may be financial toolboxes or specialized DNA toolboxes. 15

Speedy

Matlab is just way too slow. Matlab itself was built upon Java. Also Java was built upon C. So when we run a Matlab program, our computer gets busy trying to interpret and compile all that complicated Matlab code. Then it is turned into Java, and finally executes the code.

If we use C/C++, we don't waste all that time. We directly provide machine language code to the computer, and it gets executed. So ultimately we get more image processing, and not more interpreting.

After doing some real time image processing with both Matlab and OpenCV, we usually got very low speeds, a maximum of about 4-5 frames being processed per second with Matlab. With OpenCV however, we get actual real time processing at around 30 frames being processed per second.

Sure we pay the price for speed – a more cryptic language to deal with, but it's definitely worth it. We can do a lot more, like perform some really complex mathematics on images using C and still get away with good enough speeds for your application.

Efficient

Matlab uses just way too much system resources. With OpenCV, we can get away with as little as 10MB RAM for a real-time application. Although with today's computers, the RAM factor isn't a big thing to be worried about. However, our drowsiness detection system is to be used inside a car in a way that is non-intrusive and small; so a low processing requirement is vital.

Thus we can see how OpenCV is a better choice than Matlab for a real-time drowsiness detection system.

Hardware Specification:-

Processor	:	Intel Processor IV and more
RAM	:	1GB
Hard disk	:	160 GB
Web Cam		

Software Specification

Operating System	:	Windows XP/2000
Technologies used	:	.NET
Database :		MS SQL Server 2005
Software	:	Visual Studio 2008

LITERATURE REVIEW

Techniques for Detecting Drowsy Drivers

Possible techniques for detecting drowsiness in drivers can be generally divided into the following categories: sensing of physiological characteristics, sensing of driver operation, sensing of vehicle response, monitoring the response of driver.

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

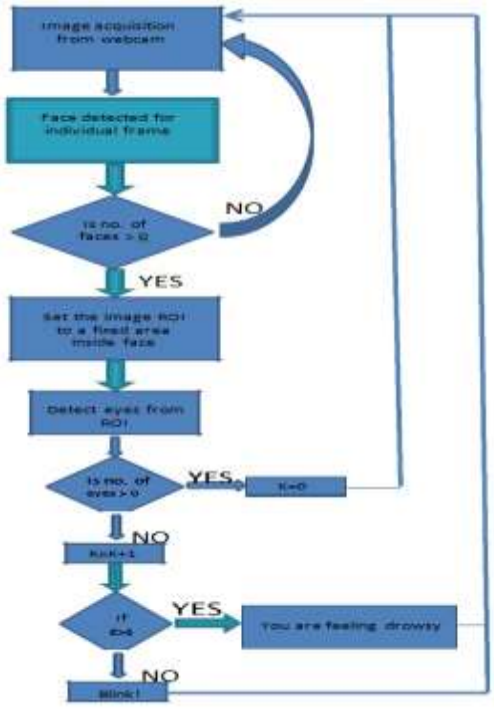
Monitoring Physiological Characteristics

Among these methods, the techniques that are best, based on accuracy are the ones based on human physiological phenomena. This technique is implemented in two ways: measuring changes in physiological signals, such as brain waves, heart rate, and eye blinking; and measuring physical changes such as sagging posture, leaning of the driver's head and the open/closed states of the eyes . The first technique, while most accurate, is not realistic, since sensing electrodes would have to be attached directly onto the driver's body, and hence be annoying and distracting to the driver. In addition, long time driving would result in perspiration on the sensors, diminishing their ability to monitor accurately. The second technique is well suited for real world driving conditions since it can be non-intrusive by using optical sensors of video cameras to detect changes.

Other Methods

Driver operation and vehicle behaviour can be implemented by monitoring the steering wheel movement, accelerator or brake patterns, vehicle speed, lateral acceleration, and lateral displacement. These too are non-intrusive ways of detecting drowsiness, but are limited to vehicle type and driver conditions. The final technique for detecting drowsiness is by 12 monitoring the response of the driver. This involves periodically requesting the driver to send a response to the system to indicate alertness. The problem with this technique is that it will eventually become tiresome and annoying to the driver.

EXISTING ALGORITHM



Flowchart

Drowsiness of a person can be measured by the extended period of time for which his/her eyes are in closed state. In our system, primary attention is given to the faster detection and processing of data.

The number of frames for which eyes are closed is monitored. If the number of frames exceeds a certain value, then a warning message is generated on the display showing that the driver is feeling drowsy.

In our algorithm, first the image is acquired by the webcam for processing. Then we use the Haarcascade file face.xml to search and detect the faces in each individual frame. If no face is detected then another frame is acquired. If a face is detected, then a region of interest in marked within the face. This region of interest contains the

eyes. Defining a region of interest significantly reduces the computational requirements of the system. After that the eyes are detected from the region of interest by using Haarcascade_eye.xml.

If an eye is detected then there is no blink and the blink counter K is set to '0'. If the eyes are closed in a particular frame, then the blink counter is incremented and a blink is detected. When the eyes are closed for more than 4 frames then it is deducible that the driver is feeling drowsy. Hence drowsiness is detected and an alarm sounded. After that the whole process is repeated as long as the driver is driving the car.

IMPLEMENTATION

Image acquisition:-

The function cvCaptureFromCAM allocates and initialized the CvCapture structure for reading a video stream from the camera.

CvCapture* cvCaptureFromCAM(int index);

Index of the camera to be used. If there is only one camera or it does not matter what camera to use , -1 may be passed.

cvSetCaptureProperty Sets camera properties For example

cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, 280);

cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, 220);

The function cvQueryFrame() grabs a frame from a camera or video file, decompresses it and returns it. This function is just a combination of *GrabFrame* and *RetrieveFrame*, but in one call. The returned image should not be released or modified by the user. In the event of an error, the return value may be NULL.

Face detection:-

Cascade is loaded by:
cascade=(CvHaarClassifierCascade*)cvLoad(cascade_name, 0, 0, 0);
storage is allocated:
CvMemStorage*storage=cvCreateMemStorage(0);
CvSeq* cvHaarDetectObjects(const CvArr* image, CvHaarClassifierCascade* cascade, CvMemStorage* storage,double scale_factor = 1.1, int min_neighbors = 3, int flags = 0, CvSize min_size = cvSize(40,40));
CvArr image is a grayscale image. If region of interest (ROI) is set, then the function will respect that region. Thus, one way of speeding up face detection is to trim down the image boundaries using ROI. The classifier cascade is just the Haar feature cascade that we loaded with cvLoad() in the face detect code. The storage argument is an OpenCV ―work buffer‖ for the algorithm; it is allocated with cvCreateMemStorage(0) in the face detection code and cleared for reuse with cvClearMemStorage(storage). The cvHaarDetectObjects() function scans the input image for faces at all scales. Setting the scale_factor parameter determines how big of a jump there is between each scale; setting this to a higher value means faster computation time at the cost of possible missed detections if the scaling misses faces of certain sizes. The min_neighbors parameter is a control for preventing false detection. Actual face locations in an image tend to get multiple ―hits‖ in the same area because the surrounding pixels and scales often indicate a face. Setting this to the default (3) in the face detection code indicates that we will only decide a face is present in a location if there are at least three overlapping detections. The flags parameter has four valid settings, which (as usual) may be combined with the Boolean OR operator. The first is CV_HAAR_DO_CANNY_PRUNING. Setting flags to this value causes fl at regions (no lines) to be skipped by the classifier. The second possible flag is CV_HAAR_SCALE_IMAGE, which tells the algorithm to scale the image rather than the detector (this can yield some performance advantages in terms of how memory and cache are used). The next flag option, CV_HAAR_FIND_BIGGEST_OBJECT, tells OpenCV to return only the largest object found (hence the number of objects returned will be either one or none).* The final flag is CV_HAAR_DO_ROUGH_SEARCH, which is used only with CV_HAAR_FIND_BIGGEST_OBJECT. This flag is used to terminate the search at whatever scale the first candidate is found (with enough neighbors to be considered a ―hit‖). The final parameter, min_size, is the

smallest region in which to search for a face. Setting this to a larger value will reduce computation at the cost of missing small faces.

```
CvRect *face = (CvRect*)cvGetSeqElem(face, 0);
```

The above function gets a sequence of objects from the image given and stores them as a rectangular region in the image.

```
cvRectangle(img,cvPoint(face->x, face->y), cvPoint(
face->x + face->width, face->y + face->height ),CV_RGB(0, 255, 0), 1, 8, 0)
```

The above function draws a rectangle in the image for the corresponding corner points .the other parameter are for drawing colour and thickness and type of lines in the rectangle.

Set Image Region of interest:-

```
cvSetImageROI(img,        /* the source image */

cvRect(face->x,   /* x = start from leftmost */

face->y + (face->height)/5, /* y = a few pixels from the top */

face->width,/* width = same width with the face */

(face->height)/3  /* height = 1/2 of face height */)
```

It sets the ROI for the corresponding image. We have taken from the left most point in the face to a few pixels from the top to half of face height. Width of the region is same as that of the face. The rectangular region is now used to get eyes.

Eye Detection:-

```
CvSeq *eyes = cvHaarDetectObjects( img,cascade, storage,1.1,5,0 /*CV_HAAR_DO_CANNY_PRUNNING*/, cvSize( 10, 5 ) );
```

It is the same function as that of the face detection .Here eye cascade is used and minimum size of the object is decreased and optimized to detect eyes of various sizes in the image.

As described earlier cvRectangle() is used to draw rectangle for sequences of eyes detected in a given frame .then cvResetImageROI(img) is used to reset the ROI so that the whole image can be used to be displayed in a window using cvShowImage("video", img) function.

Since the cascade is constructed for only open eyes ,so when eyes are closed even for momentarily they are not detected. The closing state of eyes are detected as blink and if the closing state continues for more than 7 consecutive frames then drowsiness condition is displayed using cvPutText() function . We can also use batch file command system(―abc.mp3‖) to play any audio file at the instant drowsiness is detected .

The detection process is continued until a key is pressed .When the key is pressed the program stops executing the detection function and stops capture form camera, releases memory and closes video window using the functions given below.

```
cvReleaseCapture( &capture ); cvDestroyWindow( "video" ); cvReleaseMemStorage( &storage );
```

RESULTS AND DISCUSSION

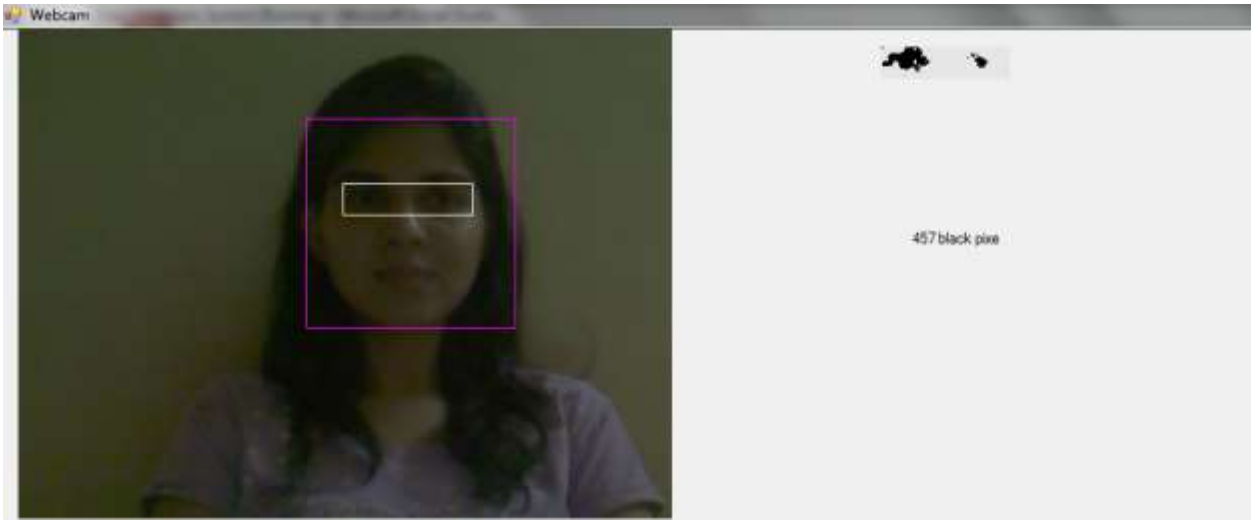
To obtain the result a large number of videos were taken and their accuracy in determining eye blinks and drowsiness was tested.

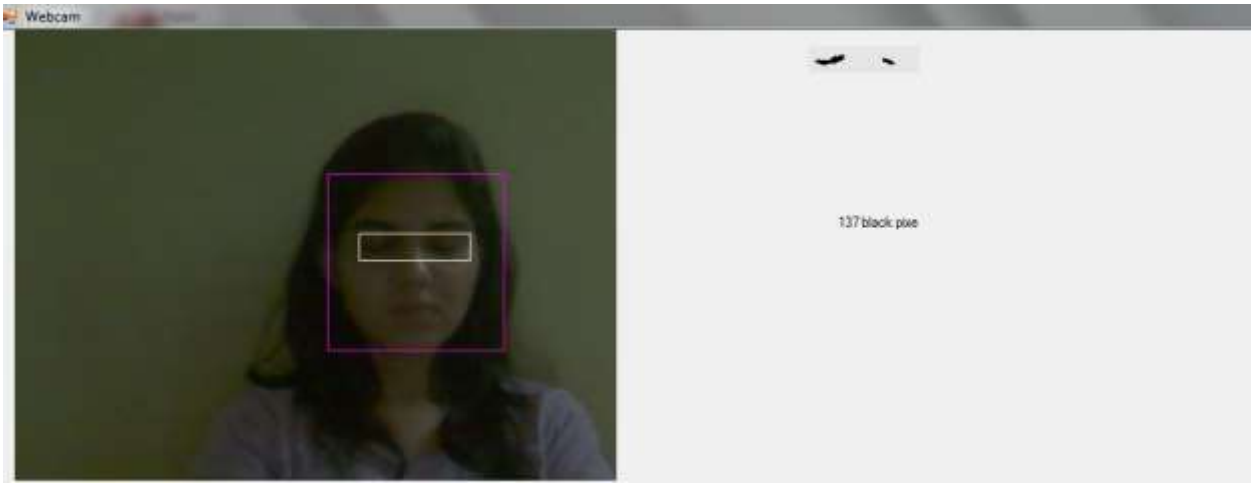
For this project we used a 5 megapixel USB webcam connected to the computer. The webcam had inbuilt white LEDs attached to it for providing better illumination. In real time scenario, infrared LEDs should be used instead of white LEDs so that the system is non-intrusive. An external speaker is used to produce alert sound output in order to wake up the driver when drowsiness exceeds a certain threshold.

When the webcam backlight was turned ON and the face is kept at an optimum distance, then the system is able to detect blinks as well as drowsiness with more than 95% accuracy. This is a good result and can be implemented in real-time systems as well.

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

Snapshot





CONCLUSION

Thus we have successfully designed a prototype drowsiness detection system using OpenCV software and Haar Classifiers. A real-time, completely non-intrusive and cost effective system that focus on the driver to detect drowsiness. We have developed a simple system consisting of modules namely image acquisition, dividing into frames, face detection, eye detection, drowsiness detection, and alerting through an alarm. Our paper can be used to detect drowsiness and thus help reducing the number of road accidents drastica lly. The system so developed was successfully tested, its limitations were identified and a future plan of action developed.

ACKNOWLEDGEMENT

It gives us a great pleasure to express our deep sense of gratitude to our guide Assistant Prof. Smita Jawale for her valuable support and encouraging mentality throughout our ongoing project. We are highly obliged to her successful progress of our Project. Our special thanks is going to Head of the Department of Computer Engineering of our college, Dr. Swapna Borde and to all of the faculties for allowing us to come here and encouraging us constantly to work hard in this project. We are highly grateful to the Honorable Principal of VCET, Dr. A. V. Bhonsale for giving

us this golden opportunity to be a part of this organization for this period**.**

REFERENCES

[1] http://www.ee.ryerson.ca/~phiscock/thesis/drowsy-detector/drowsy-detector.pdf
[2] http://www.cnblogs.com/skyseraph/archive/2011/02/24/1963765.html
[3] http://www.scribd.com/doc/15491045/Learning-OpenCV-Computer-Vision-with-the- OpenCV-Library
[4] http://opencv.willowgarage.com/documentation/reading_and_writing_images_and_vid eo.html
[5] http://www.scribd.com/doc/46566105/opencv
[6] Learning OpenCV by Gary Bradski and Adrian Kaehler
[7] http://note.sonots.com/SciSoftware/haartraining.html