A Real Time Improved Driver Fatigue Monitoring System

V. KRISHNASREE[1], N.BALAJI[2], P. SUDHAKAR RAO[3]

1. Dept of ECE, VNR Vignana Jyothi Institute of Engg &Tech, Kukatpally, Hyderabad-500090, Andhra Pradesh, INDIA kkrishnasree@yahoo.co.in
2. Dept of ECE, Jawaharlal Nehru Technological University Kakinada, Dwarapudi, Viziayanagaram-535002, Andhra pradesh, INDIA. narayanamb@rediffmail.com
3. Dept of ECE, Vignan Institute of Technology and Science, Deshmukh village, Hyderabad-508284, Andhra Pradesh, INDIA. sparvatha@gmail.com

Abstract :- One of the main technical goals in the automotive industry is to provide and increase vehicle safety. The traffic accidents are increasing day by day due to a diminished driver's vigilance level and it became a serious problem for the society. By monitoring the fatigue of the driver, the vehicle safety can be improved. Eye detection is an important initial step in Driver Fatigue Detection System. This feature can be used for developing 'Driver fatigue detection system' by monitoring attention or drowsiness of the driver. Robust non-intrusive eye tracking is a crucial step for vision based man-machine interaction technology which is widely accepted in common environments. Eye detection and tracking is an integral part of attentive user interfaces. To detect human eyes, face has to be detected initially. This is done using Open CV face Haar-cascade classifier. After obtaining successful face detection, the location of the eyes is estimated and eye detection is performed using eye haar-cascade classifier. The work aims at development of a driver safety system with visual aid to prevent accidents. The vehicle is equipped with USB web camera interfaced to the system. Here the camera is used for tracking the eyes of driver to detect fatigue. The driver assistance system unit consists of a BEAGLE BOARD with a DM3730 processor in it. The system runs on an embedded operating system called Angstrom. The system architecture deals with the development of device driver for USB web camera and DM3730 processor in standard Linux kernel. Open CV package is installed to interface USB web camera with the navigation system. Open CV package consists of various image processing functions, which are useful for identification of driver fatigue. Open CV is used as a platform to develop a code for eye detection in real time. The code is then implemented on the Beagle board (Ported with Angstrom Operating System) installed with Open CV software.

Keywords:- Driver fatigue, accident aversion, eye detection, Haar classifier and OMAP processor

1. Introduction

A great deal of computer vision research is dedicated to the implementation of systems designed to detect user movements and facial gestures. In many cases such systems are created with the specific goal of providing a way for people with disabilities or limited motor skills to be able to use computer systems. Many of the road accidents are occurring due to driver fatigue that is driver drowsiness or driver sleepiness. Sleepiness reduces the concentration, activeness, alertness and vigilance of the driver and it makes the driver to take slow decisions and sometimes no decision. Drowsiness affects the mental alertness and decreasing the driver ability to operate a vehicle safely and increasing the risk of human error that could lead to fatalities and injuries. The reasons for the fatigue related crashes are long journeys on monotonous roads, driving after eating or taking an alcoholic drink, having less sleep than normal, after taking medicines that cause drowsiness, driving after long working hours and journeys after night shifts etc. Hence there is a need to address this problem to avoid accidents by alerting the driver so that road safety can be increased. Drowsy drivers exhibit some observable behaviors such as eye gaze, eyelid movement, head movement, yawning brain wave activity and heart beat etc.

The noninvasive computer vision systems monitor these bio behavioral physical behaviors such as eye gaze, eye closure, pupil occlusion and invasive computer vision systems monitor brain wave activity and heart beat etc. So driver assistance systems must acquire, interpret and feedback the information to

warn the driver and control the vehicle. Eyes are one of the most salient features of the human face, and they play an important role in interpreting and understanding a person's desires, needs and emotional states. In addition the unique geometric, photometric and motion characteristics of the eyes also provide important visual cues for face detection, recognition and for understanding facial expression. Saito et al proposed a vision system to detect physical and mental conditions of the driver from the line of sight (gaze) [2]. Boverie et al described a system for monitoring driving vigilance by studying the eye lid movement [5]. C. Lavergne et al proposed a technology to detect the behavior of the driver by monitoring the transportation hardware systems under the control of the driver such as steering, wheel moments, breaks and gears etc. [1]. Ishi et al and R.Onken developed an active real time image based driver fatigue monitoring system [6,4]. Uneo et al described a driver drowsiness detection system based on eyes opening and closing by computing the degree of eye openness [3]. In this work a driver safety system is developed with visual aid to prevent accidents. The vehicle was equipped with USB web camera interfaced to the system for tracking the eyes of driver to detect fatigue. The driver assistance system unit consists of a Beagle Board with a DM3730 processor in it. The system runs on an embedded operating system called Angstrom. The system architecture deals with the development of device driver for USB web camera and DM3730 processor in standard Linux kernel using OpenCV package with the navigation system. Template matching technique was used to detect whether the eye is open or closed. The system generated an alarm over a period of time if the driver was detected with a high eye closure rate.

2. Materials

The Beagle Board is a DM3730 platform. Few of the various usage scenarios of the Beagle Board are shown in figure.1. It has been equipped with a minimum set of features to allow the user to experience the power of the DM3730. By utilizing standard interfaces, the Beagle Board is highly extensible to add many features and interfaces.

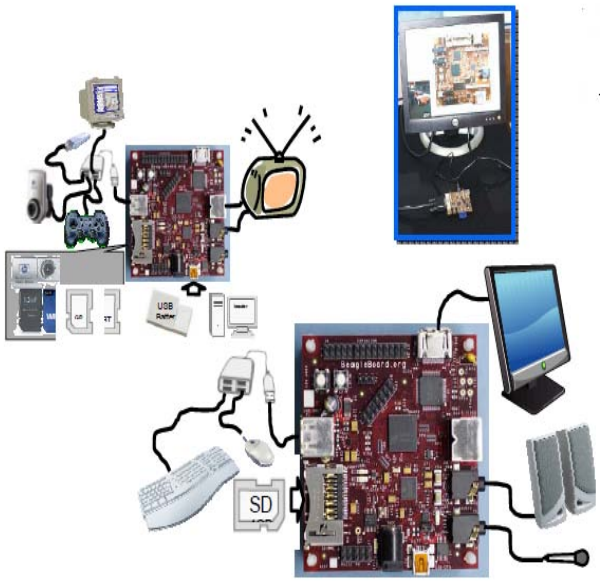

Figure.1 Beagle Board Usage Scenarios

Beagle board high level diagram is shown in the figure 2.

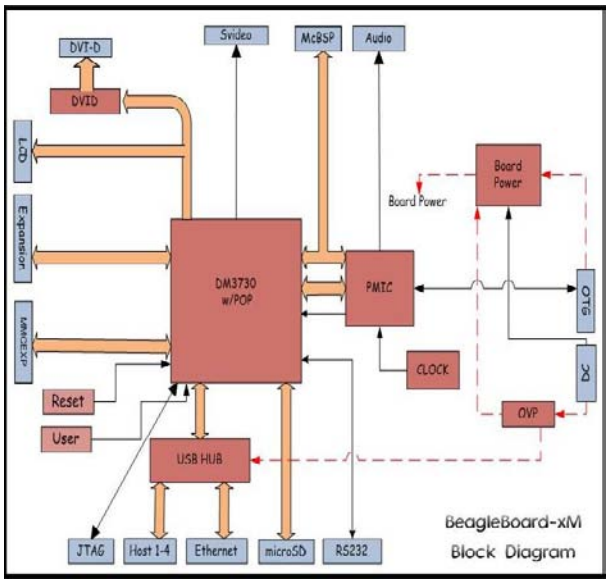

Figure.2 Beagle Board High Level Block Diagram

The high level block diagram consist of DM3730 processor with SVideo connector, DVI-D connector, Stereo In & Out connector, HS USB Host port, SD/MMC connector, JTAG, LCD, Expansion pins, Reset & User buttons.

2.1 OMAP Processor

The Beagle Board uses the DM3730CBP 1GHZ version and comes in a 4mm pitch POP package. POP (Package on Package) is a technique where the memory, NAND of 256 MB and SDRAM (Synchronous dynamic random access memory) of 512MB, are mounted on top of the DM3730. The location of key components on the Beagle Board is given in figure.3

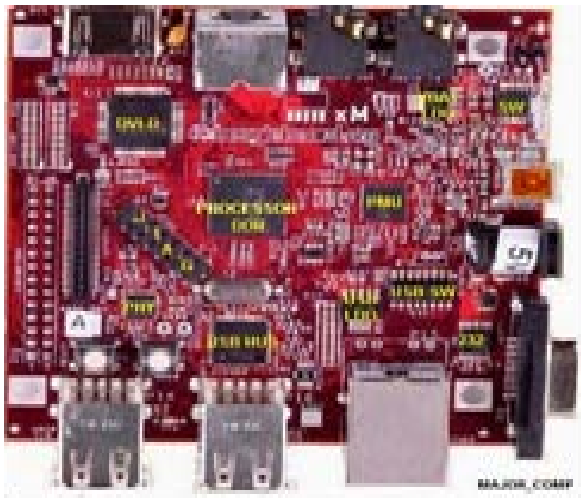

Figure.3 The location of key components on the Beagle Board.

The heart of Beagle Board is the DM3730 processor. A high level block diagram of the DM3730 is shown in figure.4. The DM3730 is a high-performance multimedia application device based on the enhanced OMAP™ 3 architecture and is integrated on TI's advanced 65-nm process technology. The DM3730 architecture is configured with different sets of features in different tier devices. The DM3730 supports high-level operating systems such as Windows CE, Linux, QNX and Symbian .This DM3730 device includes state-of-the-art power-management techniques required for high-performance low power products. The DM3730 supports the following functions and interfaces on the Beagle Board: Microprocessor unit (MPU) subsystem based on the ARM Cortex- A8™ microprocessor, POP Memory interface, Gb MDDR (256Mbytes), Gb NAND Flash (256Mbytes), 24 Bit RGB Display interface (DSS), SD/MMC interface, USB OTG interface, NTSC/PAL/S-Video output, Power management, Serial interface, I2C interface, I2S Audio interface (McBSP2), Expansion McBSP1, JTAG debugging interface.

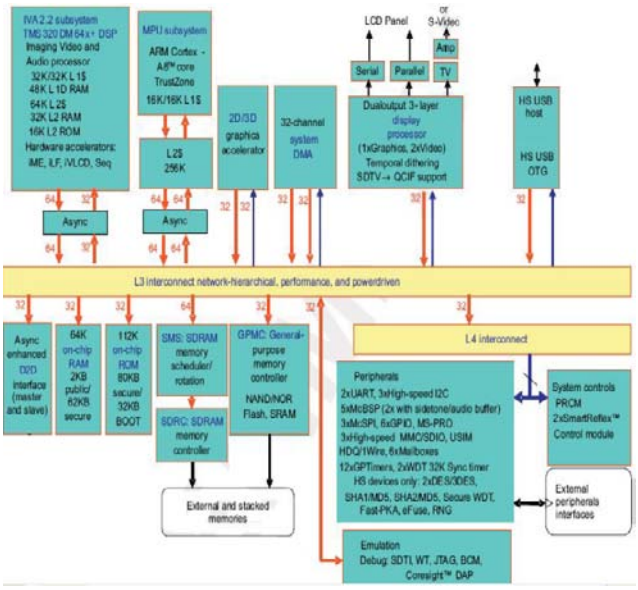

Figure.4 DM3730 Block Diagram

3. Method

3.1 Camera Interface

A device driver is a collection of subroutines and data within the kernel that constitutes the software interface to an input device. When the kernel recognizes that a particular action is required from the device, it calls the appropriate driver routine, which passes control from the "user process" to the driver routine. Control is returned to the user process when the driver routine has completed. A device driver is essentially a kernel component, but is developed independently from the rest of the kernel. Therefore interfacing is provided between the driver and the host operating system. A standard implementation would have two interfaces. The first one would communicate with the hardware itself and the second one will communicate with the operating system and the user. In Linux there exists a subsystem called "USB Core" with an application program interface (API) to support USB devices and the host controllers. Its purpose is to abstract all hardware or device dependent parts by defining a set of data structures, macros and functions. The USB core contains routines common to all USB device drivers and host controller drivers. These functions can be grouped into an upper and a lower API layer. Camera interface is given in figure.5

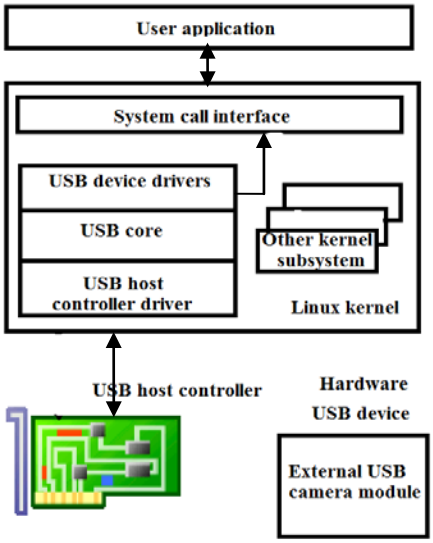

Figure.5 Design overview of camera interface

A Linux USB driver needs to register itself with the Linux USB subsystem, giving it some information about which devices the driver supports and which functions to call when a device supported by the driver is inserted or removed from the system. All of this information is passed to the USB subsystem. When the driver is unloaded from the system, it needs to unregister itself with the USB subsystem. When a device is plugged into the USB bus that matches with the device pattern of the driver registered with the USB core, the respective function is called and the USB device structure. The interface number and the interface ID are passed to the function. The driver now verifies that this device is actually one that it can accept. If so, it returns '0', otherwise if any error occurs during initialization, an error code is returned from the function. Now that the device is plugged into the system and the driver is bound to the device, any of the functions in the file operations structure that were passed to the USB subsystem will be called from a user program trying to communicate with the device. The first function called will be opened as the program tries to open the device for I/O. After the open function is called, the read and write functions are called to receive and send data to the device. In the write function, we receive some data that the user wants to send to the device. The function determines how much data it can send to the device based on the size of the write function it has created (this size depends on the size the device has). Then it copies the data from user space to kernel space, by pointing to the data and submitting to the USB subsystem. One of the most difficult problems that USB drivers must be able to handle smoothly is the fact that the USB device may be removed from the system at any point in time, even if a program is currently working on it. It needs to be able to shut down any current reads, writes and notify the user-space programs that the device is no longer there for every read, write, release and other functions that expect a device to be present, the driver first checks to see if the device is still present. If not, it realizes that the device has disappeared, and an error is returned to the user-space program. When the release function is eventually called it does the cleanup function. In a nutshell, V4L2 abstracts different video devices behind a common API that applications can use to retrieve video data without being aware of the particularities of the involved hardware. The Linux UVC driver, or short uvc video, is a Video4Linux 2 and a USB driver at the same time. It registers with the USB stack as a handler for devices of the UVC device class and, whenever a matching device is connected, the driver Initializes the device and registers it as a V4L2 device.

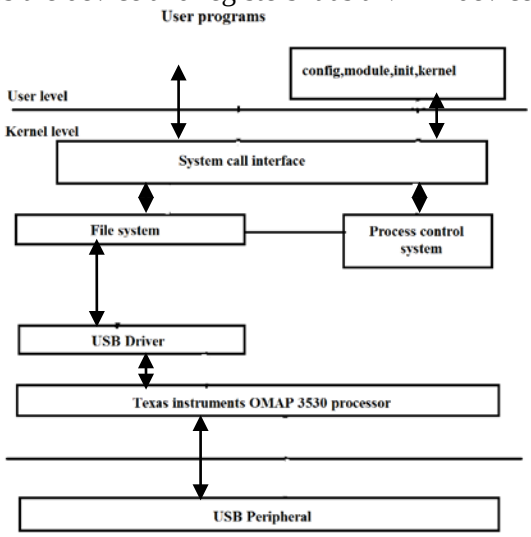

Figure.6 Functional block diagram.

The functional block diagram given in figure.6 primarily illustrates two levels, user level and kernel level. So the USB driver will be communicating with other functions as shown in the figure 2.1 at these levels. At user level user programs are written according to application and communicated with the kernel level through system call interface which communicates with the USB driver in DM3730 processor. The USB device is connected to the USB peripheral on beagle board.

3.2 Developing an Embedded Environment In Linux With Angstrom Operating System

Linux for the OMAP Software Development Platform was built by installing an Open Source tool chain and building an Open Source Linux kernel. UBUNTU9.10 Linux was installed on the PC used to build the kernel for this work. We had built the line kernel on the beagle board.

3.3 Real time eye tracking and fatigue detection

The proposed system detects the user's eye blinks and analyzes the pattern and duration of the blink to detect fatigue of driver. Eye detection is an important initial step in Driver Fatigue Detection System .This feature can be used for developing Driver Fatigue Detection System by monitoring attention or drowsiness of the driver. After the automatic initialization of the system occurs from the processing of the user's involuntary eye blinks in the first few seconds of use, the eye is tracked in real time using correlation with an online template. If the user's depth changes significantly or rapid head movement occurs, the system is automatically reinitialized. There are no lighting requirement or offline templates needed for the proper functioning of the system. The system works with inexpensive USB camera and runs at a frame rate of 30 frames per second. Extensive experiments were conducted to determine both the system's accuracy in classifying voluntary and involuntary blinks, as well as the system's fitness in varying environment conditions, such as alternative camera placements and different lighting conditions. The automatic initialization phase is triggered by the analysis of the involuntary blinking of the current user of the system, which creates an online template of the eye to be used for tracking. This phase occurs each time the current correlation score of the tracked eye falls below a defined threshold in order to allow the system to recover and regain its accuracy in detecting the blinks.

3.3.1 Working Algorithm:

The algorithm for detecting and analyzing blinks is initialized automatically, dependent only upon the inevitability of the involuntary blinking of the user. Motion analysis techniques are used in this stage, followed by online creation of a template of the open eye to be used for the subsequent tracking and template matching that is carried out at each frame. A flow chart depicting the main stages of the system is shown in Figure.8.

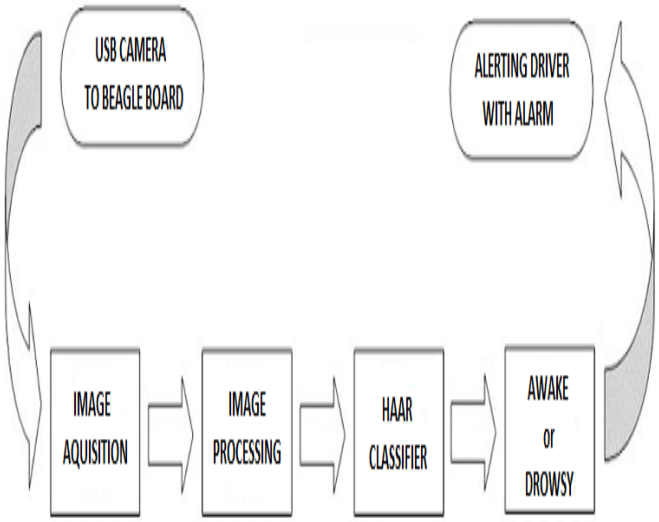

Figure.8 Overview of the main stages in Driver Fatigue Detection System

After the detection of eye in Image Processing block, Fatigue detection has to be done and is given in the figure.9

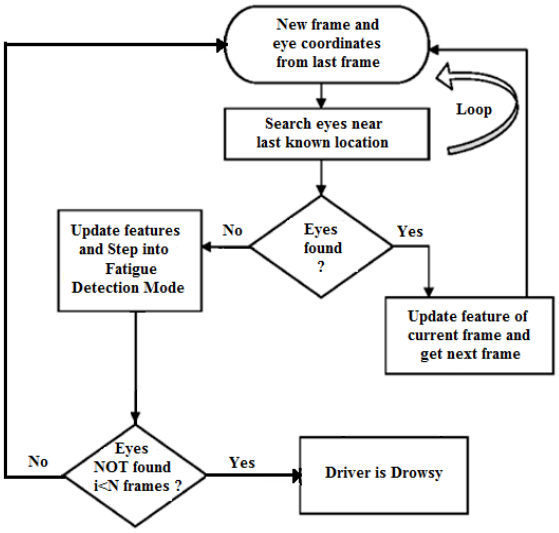

Figure.9 Eye tracking and Fatigue Detection

3.3.2 Initialization

The first step in analyzing the blinking of the user is to locate the eyes. To accomplish this, the difference image of each frame and the previous frame is created and then thresholded, resulting in a binary image showing the regions of movement that occurred between the two frames. Next, a 3x3 star-shaped convolution kernel is passed over the binary difference image in an Opening morphological operation. This functions to eliminate a great deal of noise and naturally-occurring jitter that is present around the user in the frame due to the lighting conditions and the camera resolution, as well as the possibility of background movement. In addition, this Opening operation also produces fewer and larger connected components in the vicinity of the eyes (when a blink happens to occur), which is crucial for the efficiency and accuracy of the next phase. A recursive labeling procedure is applied next to recover the number of connected components in the resultant binary image. (the left eye and the right eye). In the case that other movement has occurred, producing a much larger number of components, the system discards the current binary image and waits to process the next involuntary blink in order to maintain efficiency and accuracy in locating the eyes. Given an image with a small number of connected components output from the previous processing steps, the system is able to proceed efficiently by considering each pair of components as a possible match for the user's left and right eyes. The filtering of unlikely eye pair matches is based on the computation of six parameters for each component pair: the width and height of each of the two components and the horizontal and vertical distance between the centroids of the two components. A number of experimentally-derived heuristics are applied to these statistics to pinpoint the exact pair that most likely represents the user's eyes. For example, if there is a large difference in either the width or height of each of the two components, then they likely are not the user's eyes. As an additional example of one of these many filters, if there is a large vertical distance between the centroids of the two components, then they are also not likely to be the user's eyes, since such a property would not be humanly possible. Such observations not only lead to accurate detection of the user's eyes, but also speed up the search greatly by eliminating unlikely components immediately.

3.3.3 Template Creation

If the previous stage results in a pair of components that passes the set of filters, then it is a good indication that the user's eyes have been successfully located. At this point, the location of the larger of the two components is chosen for creation of the template. Since the size of the template that is to be created is directly proportional to the size of the chosen component, the larger one was chosen for the purpose of having more brightness information, which will result in more accurate tracking and correlation scores. Since the system will be tracking the user's open eye, it would be a mistake to create the template at the instant that the eye was located, since the user was blinking at this moment. Thus, once the eye is believed to be located, a timer was triggered. After a small number of frames elapse, which was judged to be the approximate time needed for the user's eye to become open again after an involuntary blink, the template of the user's open eye was created. Therefore, during initialization, the user is assumed to be blinking at a normal rate of one involuntary blink every few moments. Again, no offline templates are necessary and the creation of this online template is completely independent of any past templates that may have been created during the run of the system.

3.3.4 Eye Tracking

Template matching is necessary for the desired accuracy in analyzing the user's blinking since it allows the user some freedom to move around slightly. Though the primary purpose of such a system is to serve people with paralysis, it is a desirable feature to allow for some slight movement by the user or the camera that would not be feasible if motion analysis were used alone. The normalized correlation coefficient is used to accomplish the tracking. This measure is computed at each frame using the following formula:

$$\frac{\sum_{x,y}[f(x,y)-f_{u,v}][t(x-u,y-v)-t]}{\sqrt{\sum_{x,y}[f(x,y)-f_{u,v}]^2\sum_{x,y}[t(x-u,y-v)-t]^2}}$$

Where, f (x, y) is the brightness of the video frame at the point (x, y), f (u, v) is the average value of the video frame in the current search region, t(x, y) is the brightness of the template image at the point (x, y), and t is the average value of the template image. The result of this computation is a correlation score between -1 and 1 that indicates the similarity between the open eye template and all points in the search

region of the video frame. Scores closer to 0 indicate a low level of similarity, and to 1 indicate a probable match for the open eye template. A major benefit of using this similarity measure to perform the tracking is that it is insensitive to constant changes in ambient lighting conditions. Since this method requires an extensive amount of computation and is performed 30 times per second, the search region is restricted to a small area around the user's eye. This reduced search space allows the system to remain running smoothly in real time since it drastically reduces the computation needed to perform the correlation search at each frame.

3.3.5 Eye Detection

Algorithm is shown in figure.10

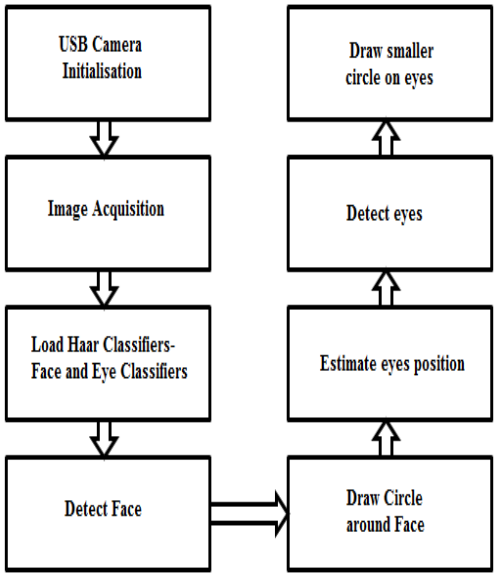

Figure.10 Eye Detection Flow Chart

Initially all the components are connected to beagle board which works as central processing unit for the whole system. USB camera is also attached to it. The camera captures the video of the person sitting in front of it .Video file is converted into frames. OpenCV face-haar classifier is loaded. Each frame is compared with the pre-defined features of the haar classifiers. When the features are matched the face is detected and a circle is drawn around the face. Using feature extraction we estimate the position of the eyes. By comparing with the OpenCV eye-haar classifier the eyes are detected and circles are drawn around them.

3.3.6 Haar Classifier

The Haar classifier is the tree-based technique in OpenCV, which builds a boosted rejection cascade. It was developed earlier as a full-fledged face-recognition application trained to recognize faces and other rigid objects. Face detection is built on the well-known and often used field of statistical boosting. This technique has been used to create state-of-the-art detectors with a different detector trained for each view or pose of the object. Thus, the Haarclassifier is a valuable tool for recognition tasks. The pre trained objects are available in OpenCV for face detection and for eye detection. A recognition process can be much more efficient if it is based on the detection of features that encode some information about the class to be detected. This is the case of Haar-like features that encode the existence of oriented contrasts between regions in the image. A set of these features can be used to encode the contrasts exhibited by a human face and their spatial relationships. Haar-like features are so called because they are computed similar to the coefficients in Haar wavelet transforms. First, a classifier is trained with a few hundreds of sample views of a particular object, called positive examples that are scaled to the same size, and negative examples that are arbitrary images of the same size. After a classifier is trained, it can be applied to a region of interest in an input image. The classifier gives output a "1" if the region is likely to show the object and "0" otherwise. To search for the object in the whole image one can move the search window across the image and check 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, rather 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. The "cascade" classifier means that the resultant classifier consists of several simpler classifiers that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. The word "boosted" means that the classifiers at every stage of the cascade are complex themselves and they are built out of basic classifiers using one of Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost boosting techniques. The basic classifiers are decision-tree classifiers with at least 2 leaves. Haar-like features are the input to the basic classifers. The feature used in a particular classifier is specified

by its shape, position within the region of interest and the scale.

3.3.7 Fatigue Detection

The detection of blinking and the analysis of blink duration are based solely on observation of the correlation scores generated by the tracking at the previous step using the online template of the user's eye. As the user's eye closes during the process of a blink, its similarity to the open eye template decreases. Likewise, it regains its similarity to the template as the blink ends and the user's eye becomes fully open again. This decrease and increase in similarity corresponds directly to the correlation scores returned by the template matching procedure. Close examination of the correlation scores over time for a number of different users of the system reveals rather clear boundaries that allow for the detection of the blinks. As the user's eye is in the normal open state, very high correlation scores of about 0.85 to 1.0 are reported. As the user blinks, the scores fall to values of about 0.5 to 0.55. Finally, a very important range to note is the one containing scores below about 0.45. Scores in this range normally indicate that the tracker has lost the location of the eye. In such cases, the system must be reinitialized to relocate and track the new position of the eye. Given these ranges of correlation scores and knowledge of what they signify derived from experimentation and observation across a number of test subjects, the system detects voluntary blinks by using a timer that is triggered each time the correlation scores fall below the threshold of scores that represent an open eye. If the correlation scores remain below this threshold and above the threshold that results in re-initialization of the system for a defined number of frames that can be set by the user, then a voluntary blink is judged to have occurred, causing a mouse click to be issued to the operating system.

4. Experiments and Results

The hardware is setup and experimental trails are conducted in two automobile vehicles made by Tata and Benz. Total of four people with two Male and two Female participated in the experimental trails.

We used Open CV as a platform to develop a code for eye detection in real time. The code is then implemented on the Beagle board (Ported with Angstrom Operating System) installed with Open CV software. To detect human eyes, face has to be detected initially. This is done using OpenCV face haarcascade classifier. After obtaining successful face detection, the location of the eyes is estimated and eye detection is performed using eye Haar-cascade classifier. Input is acquired from the webcam and connected to the beagle board. The beagle board is installed with open CV and Haar classifiers. Hence using the open CV, face and eyes are detected accurately and displayed on the DVI monitor as shown in the figure. 11 (a). The larger circle indicates the face while smaller circle indicates the eyes. When eyes are closed only the face is detected. The blue circle indicates the person. Figure.11 (b) shows the detected face. Eye detection is obtained for people wearing spectacles shown in figure.12.

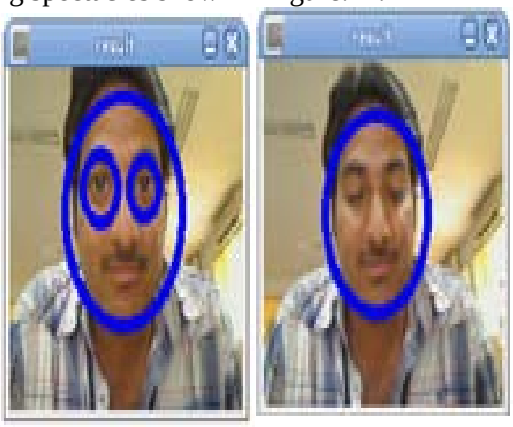

Figure.11 (a) Photographs showing the detected face and eyes. (b) When eyes are closed only the face is detected.

Figure.12 Photograph showing the detected faces and eyes for person with spectacles.

ANGSTROM Operating System with OPEN EMBEDDED and BITBAKE was ported on to the Beagle Board. A uImage was built by using Linux kernel which is compatible with OMAP and kernel configuration and compiling are done. ANGSTROM Operating System was ported on to the Beagle Board with the help of µSD card is shown in the figure.13 Figure.13(b) shows Log-in window on DVI-monitor after porting Operating System on to the Beagle Board.

Figure.13 (a) Beagle board with µSD card
Figure.13 (b) Login window

Given the reasonable assumption that the user is positioned anywhere from about 1 to 2 feet away from the camera, the eyes were detected within moments in all of the experiments in which the subjects were seated between 1 and 2 feet from the camera, it never took more than three involuntary blinks by the user before the eyes were located successfully. The real time eye tracking and fatigue detection were performed. When the person was drowsy it tracked the eye and detected the fatigue and warned the driver about the drowsiness by alerting the driver with the alarm and also by the text as shown. Whenever the driver felt drowsy, his eyes tend to close. If they remain closed for some time, then the alarm was generated as a warning to the driver. The following figure.14 shows the fatigue of the driver and the alarm was played in the background. When the alarm was generated, it alerted the driver with a warning message, "Pay Attention..!!!" . It is shown in the figure.15

Figure.14 Eye tracking and Fatigue Detection

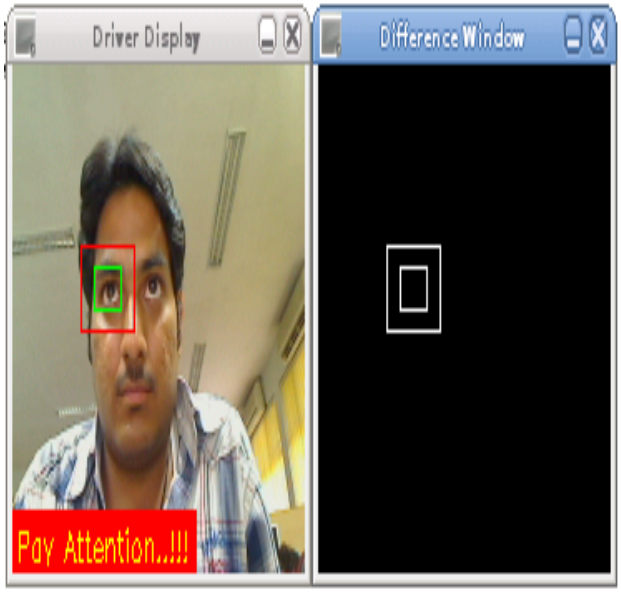

Figure.15 Alerting the driver by showing the warning message

5. Conclusions

The system proposed in this analysis provides accurate detection of driver fatigue. The automatic initialization phase is greatly simplified with no loss of accuracy in

locating the user's face and eyes and choosing a suitable open eye template. Given the reasonable assumption that the user is positioned anywhere from about 1 to 2 feet away from the camera, the face and eyes are detected within moments and it never took more than three involuntary blinks by the user before the eyes were located successfully. As the distance increases beyond this amount, the face and eyes can still be detected in some cases, but it may take a longer time to occur since the candidate pairs are much smaller and start to fail the tests designed to pick out the likely components that represent the user's eyes. Higher frame rates and finer camera resolutions could lead to more robust eye detection that is less restrictive on the user, while increased processing power could be used to enhance the tracking algorithm to more accurately follow the user's eye and recover more gracefully when it is lost. The ease of use and potential for rapid input that this system provides could be used to enhance productivity by incorporating it to generate input for a task in any general software program. This work is able to accurately detect fatigue and also prompt the driver through alarm for preventing accidents; the audio is synchronized with the camera for alerting purpose and display the output.

References

[1] C. Lavergne, P. De Lepine, P. Artaud, S. Planque. : Results of the feasibility study of a system for warning of drowsiness at the steering wheel based on analysis of driver eyelid movements. Presented at the Proc.15th Int. Tech. Conf. Enhanced Safety Vehicles, vol. 1, Melbourne, Australia, (1996).

[2] H. Saito, T. Ishiwaka, M. Sakata, and S. Okabayashi. :Applications of drivers line of sight to automobiles-what can drivers eye tell, in Proc. Vehicle Navigation Information Systems Conf., Yokohama, Japan, Aug. pp. 21-26. (1994)

[3] H. Ueno, M. Kaneda, and M. Tsukino. : Development of drowsiness detection system. in Proc. Vehicle Navigation Information Systems Conf, Yokohama, Japan, Aug. pp. 15-20 (1994).

[4] R. Onken.: Daisy, an adaptive knowledge-based driver monitoring and warning system. in Proc. Vehicle Navigation Information Systems Conf., Yokohama, Japan, Aug. pp. 3-10 (1994).

[5] S. Boverie, J. M. Leqellec, and A. Hirl. : Intelligent systems for video monitoring of vehicle cockpit. in Proc. Int. Congr. Expo. ITS: Advanced Controls Vehicle Navigation Systems. pp. 1-5 (1998).

[6] T. Ishii, M. Hirose, and H. Iwata. : Automatic recognition of driver's facial expression by image analysis. J. Soc. Automotive Eng. Japan, vol. 41, no. 12, pp. 1398-1403, (1987).