SMART Mobility Model for Driver Assistance in Semi-Autonomous Vehicles

Kezia M
Department of ECE
PSG College of Technology
Coimbatore, TN, India
1907NL05@psgtech.ac.in

Anusuya K V
Department of ECE
PSG College of Technology
Coimbatore, TN, India
kva.ece@psgtech.ac.in

Bharath M
Caterpillar India Pvt Ltd.
Chennai, TN, India
bharathdk5@gmail.com

***Abstract*— The National Highway Traffic Safety Administration database reveals that drowsy driving causes more than 100,000 vehicle crashes a year. The increase in toll percentage is due to loss of driver control, human errors, and vehicle malfunctions. A driver assistance model to lead the vehicle to the nearest safe zone is proposed to automate the driver control system. EAR and ETV are the metrics used to detect the abnormalities of the driver. Safe zones are identified using GPS and Google API keys and the nearest safe zone is selected using Haversine geographic distance formula. Further, the lane detection to support the autonomous movement of the vehicle to the selected zone is guided by the YOLO algorithm. The algorithms are coded using Python scripts and trained with a predefined, generalized public database.**

Keywords—Advanced driver assistance system (ADAS), Automatic driver control system, Drowsiness detection, YOLO algorithm.

I. INTRODUCTION

Drowsiness is a biological disorder before falling asleep. Save Life Foundation (SLF), a non-profit, private organization committed to improving road safety and emerging medical care across India reports the death of 13, 81,314 people in preventable road crashes in the last 10 years. And further, 80% of these fatal accidents are due to drowsy driving [1]. The measures employed for drowsiness detection are categorized as physiological, behavioural, and vehicle-based. Monitoring the ECG / EMG / EEG signals of the driver, while driving the vehicle is suggested as a solution for the detection of abnormalities.

Further actions to prevent accidents due to drowsiness include the generation of alarm and informing the driver through the generic Smartphone APP etc [2]. However, the proposed technique is based on behavioural measures.

II. RELATED WORK

Amna Rahman [3] presented a method for detecting the drowsiness of a person using the midpoint of the eye layers and calculated the blink rate with a high-resolution web camera (16MP) and obtained 94% accuracy under good lighting conditions and clear visibility of eyes. Moreover, the algorithm would fail, if the eyes are covered with sunglasses.

A component for the Advanced Driver Assistance System (ADAS) to automatically detect drowsiness is discussed in [4]. The module uses AI algorithms along with the visual data being captured. The system identifies and monitors the face and eyes and determines drowsiness using Support Vector Machines (SVMs). The system is designed to work under changeable light conditions in real-time. This system considers other distractions of the driver like yawning, head tilts, and face orientation along with eye blinking. These additional feature extractions improved the system's reliability. But this system suffers from a significant error percentage in generating unexpected false alarms.

Chuang-Wen et al[5] introduced "Car Safe", the first android smartphone application for drowsiness detection. The application requires a dual-camera Smartphone and operates by switching between two camera pipelines. The front camera pipelines monitor the driver's eye blinks rate and head pose to determine drowsiness. The back camera determines the vehicle's distance from other vehicles on the road. It also checks the lane change situation and has an 83% precision and 17% recall.

Sahayadhas et al., [6], used ElectroOculoGraphy(EOG) signal in 2013 to measure the cornea-retina potential difference and monitored the eye movements related to drowsiness. The measures such as ElectroEncephaloGraphy (EEG), ElectroCardioGraphy (ECG), and ElectroMyoGraphy (EMG) could also be used to improve the system efficiency. In the future, ECG and EMG signals can be combined with vision-based measures for yielding better accuracy in decision making.

III. PROPOSED WORK

The proposed model has three modules.

- Drowsiness detection through Eye-tracking
- Safe zone identification using geographical distance estimation
- Object detection for self-driving vehicles on the way to nearest safe zone.

A. Drowsiness detection through Eye-tracking

In paper [7], the safety technique algorithm demanding the continuous monitoring of the driver's eyes to detect abnormal activities is elaborated. The tracking of eyeballs involves the measurement of horizontal and vertical distances between the edges of the eyes named Euclidean Distance (ED). The Eye Aspect Ratio (EAR) is computed and compared with the threshold value of the average blink rate to determine the abnormality. The average threshold value computation that involves the survey of blink rates at different scenarios is discussed in [8]. Fig. 1 depicts the steps involved in the tracking of eyeballs to detect drowsiness. From the video stream file of the driver's face, each frame is resized and converted into grayscale. The (x,y) coordinates of the left and right eyes are extracted from the face and the EAR and eye blink threshold values are calculated for each eye. The shape predictor.dat (Dataset for eyelid outline) is the predefined library used for extraction of the shape from the video or an image. Each eye is represented by 68 numbers of (x, y) coordinates, starting from the left corner of the eye, and moving clockwise around the remaining region. The relation

- Obtain video frames of the vehicle driver's face from the camera
- Detect face frames from the video and extract eye co-ordinates
- Convert to Grayscale using Luminosity algorithm
- Compute EAR with edges of both eyes to calculate blink rate
- If distance ≈ 0, eye-state is closed.
- If the eye state is "closed" constantly ≥2, {confirm drowsy driving;

  Enable alarm};

Fig. 1. Drowsiness detection algorithm

between the width and height of these coordinates is given in (1).

$$EAR = (Verti\_ED1+Verti\_ED2)/(2*Hori\_ED) \quad (1)$$

The coordinate points of (1) are shown in (2).

$$EAR = (P2– P6) + (P3– P5) / (2*(P1 – P4)) \quad (2)$$

Where P1 to P6 are the 2D facial landmark locations shown in Fig. 2.

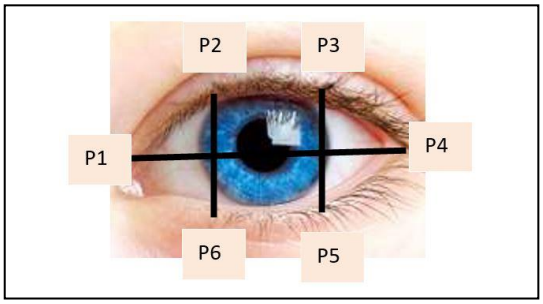


Fig. 2. Eye co-ordinates

The distance between the vertical eye landmarks and horizontal eye landmarks is computed. The ratio of eye landmark distances is used to determine the blinking status of the person. Hence, the simple equation avoids the use of image processing techniques. If an eye is fully open, the EAR would be larger and relatively constant over time. Whenever the person blinks the EAR decreases and approaches zero. The mathematical distance between the two coordinate points is called ED and is computed for vertical and horizontal eye coordinates as shown in (3).

$$ED = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (3)$$

Where, x_1, x_2, y_1 and y_2 are the co-ordinate points.

If EAR varies above and below the threshold, it is registered as a "blink" and is termed as EYE_AR_THRESH. Its optimal value is referred to as 0.24 [9].

If (EAR <EYE_AR_THRESH)

then {Eye Lid is closed}

Else {Eye lid is open}

Fig. 3. Condition for drowsiness detection

Further, two counter variables namely COUNTER and TOTAL are incorporated to ascertain the conditions for drowsiness. COUNTER is the total number of successive frames that have an EAR < EYE_AR_THRESH. TOTAL is the total number of blinks that are obtained during the run time of the Python script.

Also, EYE_AR_CONSEC_FRAME, an important constant is set to 3 to indicate that three consecutive frames with an EAR < EYE_AV_THRESH must happen for a blink to be registered. This condition is depicted in Fig. 4.

if (COUNTER>= EYE_AR_CONSEC_FRAME)

then {TOTAL = TOTAL +1}

Fig. 4. Blink detection and update

B. Safe Zone Identification using Geographical Distance Estimation

The confirmation of abnormality demands the redirection of the vehicle to a safe zone. The position of the vehicle (Source) and the safe zone places (Destination) is obtained from the GPS module and its API keys. Using GPS output, the Latitudinal and Longitudinal coordinates of the entire safe zone and the vehicle are obtained. Each place in the safe zone has a unique ID. The nearest safe zone is finalized with the shortest distance algorithm. The distance from the source and destination Latitudes and Longitudes is calculated by using the HAVERSINE method guided by (4), (5), and (6) [10]. This method remains well-conditioned for numerical computation even at small distances. Equations (4a) and (4b) calculate the hypotenuse distance between the two sets of coordinates. Equation (5) defines the angle in the Euclidean plane (2-argument arctangent) and (6) determines the distance between the two points.

$$Distance\ \_btw\_lon = Dest\_lon – Src\_lon \quad (4a)$$

$$Distance\ \_btw\_lat = Dest\_lat – Src\_lat \quad (4b)$$

$$S = \{[\sin (Distance\ \_btw\_lat/2)\ \hat{}2)] + [\cos (Src\_lat * \cos (Dest\_lat) * \sin (Distance\ \_btw\_lon\ /2)\ \hat{}2\} \quad (5)$$

$$Distance = 2*R*S*\tan^2 (\sqrt{S}, \sqrt{(1 - S)}) \quad (6)$$

Where R= Approximate Radius of Earth = 6371Km.

C. Object detection for Self-driving of vehicles

For the self-driving of the vehicle to the selected safe zone destination, the You Only Look Once (YOLO) algorithm for lane detection is applied [11]. The objective of the algorithm is to make the driverless vehicle move in the desired path through the Computer vision-based Line Follower technique. YOLO is a Convolution Neural Network (CNN) developed for object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts the bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. YOLO looks into the entire image during training and testing. Hence, it implicitly encodes contextual information about classes as well as their appearance. The flowchart in Fig. 5 illustrates the YOLO operation.

The CNN is trained using Adam optimizer with the loss parameter of Categorical Cross-Entropy (CCE). Using this

loss parameter, CNN can be trained to output a probability over C classes for each image.

$$CCE = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{J} y_j \cdot \log(\hat{y_j}) + (1\text{-}y_j)\cdot \log(1-\hat{y_j}) \quad (7)$$

Where, N = Number of training samples

Y = Actual output

Y^ = Predicted Output

The Gradients are calculated using Adam Optimizer and the weights of DCNN are obtained as Θ_t.

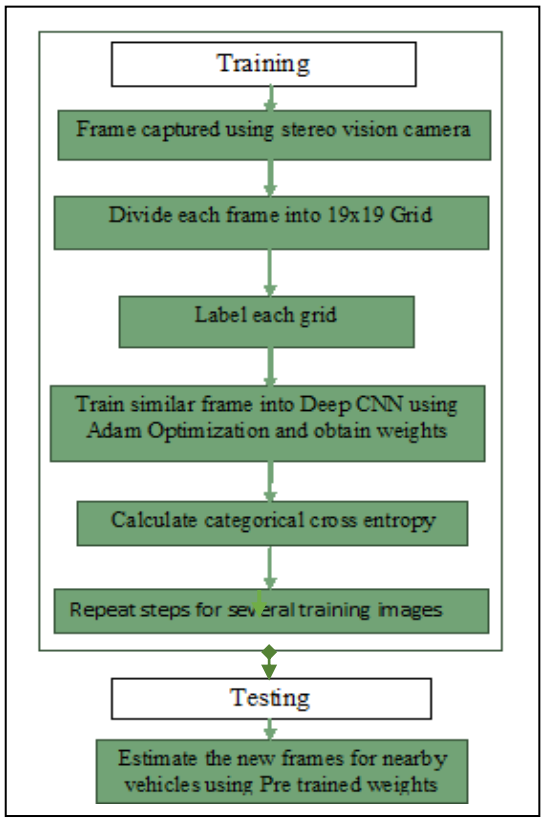


Fig. 5. YOLO flow operation

Adam – The adaptive moment estimation algorithm shown in Fig. 6, is an extension to stochastic gradient descent for deep learning applications in computer vision. It is appropriate for non-stationary objectives and problems with very noisy/sparse gradients.

$m_0 \leftarrow 0$ (Initialize first moment vector)
$v_0 \leftarrow 0$ (Initialize first moment vector)
$t \leftarrow 0$ (Initialize timestep)
While θ_t not converged **do**
$t \leftarrow t+1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$(Get gradients w.r.to stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1)\cdot g_t$ (update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2)\cdot g_t^2$ (update biased second raw moment estimate)
$\widehat{m_t} \leftarrow m_t/(1-\beta_1^{\,t})$ (compute bias corrected first moment estimate)
$\hat{v_t} \leftarrow v_t/(1-\beta_2^{\,t})$ (compute bias corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m_t}/(\sqrt{\hat{v_t}}+\epsilon)$ (update parameters)

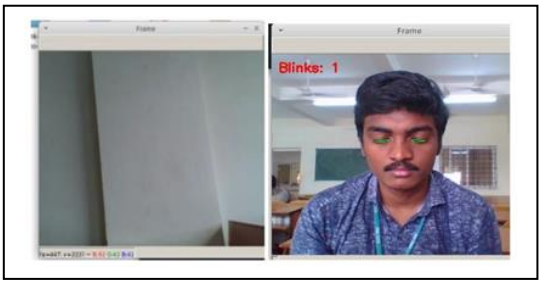
Fig. 6. Adam-optimizer algorithm

IV. Implementation and Test Results

A. Drowsiness Detection

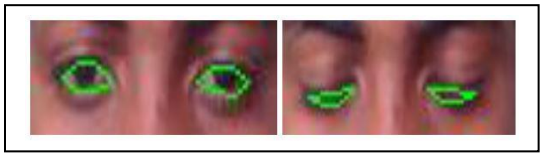
The drowsiness detection algorithm is implemented using python script. In Fig. 7a only the capture wall is shown. The human eyes are absent. Hence, there is no detection of the eyes. Therefore, the blink rate and frame counter values are calculated as zero. In Fig. 7b the face of the human is present and his eyes are detected and the number of blinks is calculated as 1.

The detection of the eye is indicated by using green lines and the Blinks = 1 is displayed. Similarly, in Fig. 7c, the eye is detected and the number of blinks is calculated and incremented whenever the conditions are met. In the first part of Fig.7d, the eyelid is above the threshold value and hence eye is open. There is no blinking that takes place. And in the second part, the eyelid is below the threshold value and hence eye is closed.



a) Absence of eye b) Detection of blink

c) Detection of the number of blinks



d)Threshold level

Fig. 7. Test Results

B. Safe zone identification

By using the Google API key - "AIzaSyBl7kIA_zo70Jnpo_zlTnWs1nzYFRvZ_KM" the information about the latitude and longitude of the safe zone is displayed. In Fig. 8, the name of the place, latitude and longitude, unique code for that place, and the distance from the source is displayed. Unique code is displayed because two safe zones can have the same name. After getting the geographical coordinates of all nearby safe zones, the distance from the source to each of them is calculated and the nearest zone is obtained. Fig. 9 displays the sample selection.

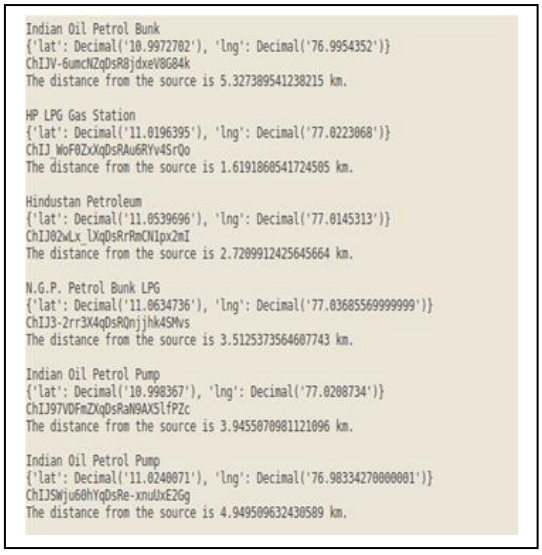


Fig. 8. Parameters of safe zone

Fig. 9. Nearest Place

C. *YOLO object detection*

In YOLO, an input image is divided into 19X19 grids and each grid is passed on to DARKNET architecture, which converts images of any size into a 7x7x1024 Tensor. The Tensor is passed on to a Fully Connected network with 4096 Neurons and its output is up sampled to give a 7x7x30 Tensor. This Tensor is then down sampled to a Tensor of size given as,

[1x (5+Total number of Classes Trained)]

The network used is trained with the COCO Dataset having 80 Classes. Here, the algorithm is used to detect 3 classes namely,

- Car
- Light
- Pedestrian

For YOLO to work properly the Training Labels have to change, instead of having a one-hot vector. It should have individual labels per grid and stack them together to obtain a label for an image.

1) *YOLO Decoding Process*: YOLO is applied to the sample input image given in Fig. 10. The YOLO architecture is shown in Fig. 11. For simplicity, the image is divided into 4X4 grids. For each grid, the following vector is obtained.

$[P_x, B_x, B_y, B_h, B_w, C_1, C_2, C_3]$

Where,

P_x -> Presence of an Object [if yes 1, else 0]

B_x, B_y, B_h, B_w -> Bounding Box Co-Ordinates

C_1 -> Presence of Class 1(Car) [if yes 1, else 0]

C_2 -> Presence of Class 2(Light) [if yes 1, else 0]

C_3 -> Presence of Class 3 (Pedestrian)[if yes 1else 0]

Hence, 16 labels are obtained for a single image divided into 4X4 grids. Stacking them one on top of each other obtains an 8X16 vector. Then, DCNN has trained to output an 8X16 vector for the given input image. Categorical Cross Entropy is used for the calculation of error. The output is shown in Fig. 12.

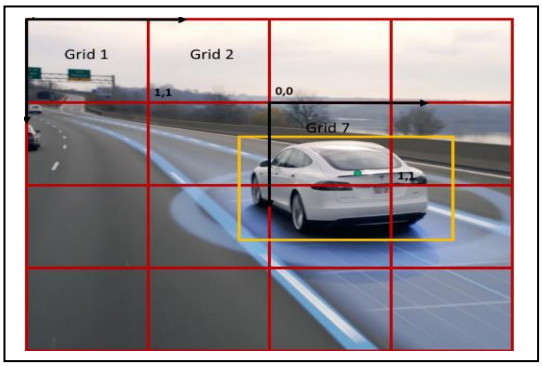


Fig. 10. Sample input image

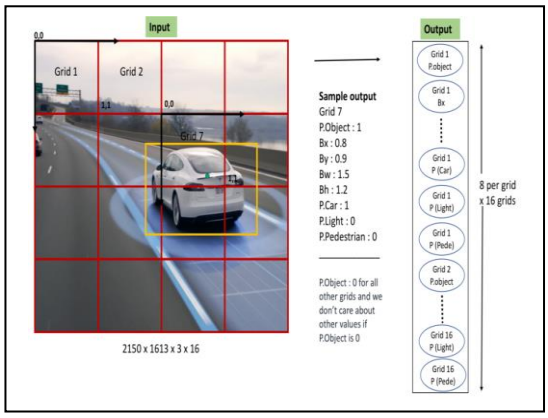


Fig. 11. YOLO algorithm Architecture

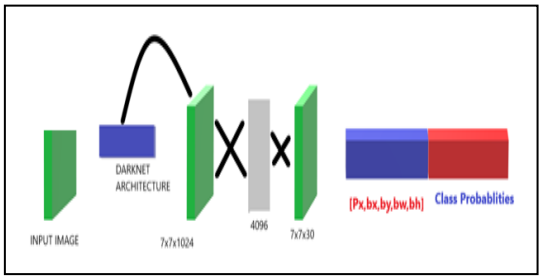


Fig. 12. Input- output relation of YOLO process

V. CONCLUSION

Through the implementation of the proposed algorithm, accidents due to drowsy driving could be prevented. By detecting the abnormalities of the driving person, the vehicle is led to the nearest safe zone. The limitation exists in detecting the abnormality of the driver. Differentiating the reason for drowsiness such as consumption of alcohol or any medical emergency would be carried out in the future.

REFERENCES

[1] https://savelifefoundation.org/

[2] M. T. Tombeng, H. Kandow, S. I. Adam, A. Silitonga and J. Korompis, "Android-Based Application To Detect Drowsiness When Driving Vehicle," 2019 1st International Conference on Cybernetics and Intelligent System (ICORIS), pp. 100-104, 2019.

[3] AmnaRahman, MehreenSirshar, Aliya Khan, “Real-Time Drowsiness Detection using Eye Blink Monitoring”, In 2015 National Software Engineering Conference (NSEC 2015).

[4] M. J. Flores, J. M. Armengol, and A. Escalera, ”Real-time warning system for driver drowsiness detection Using Visual Information”, Journal of Intelligent & Robotic Systems, Volume 59, Issue 2, pp 103-125, August 2010.

[5] C. W. You, N. D. Lane, F. Chen, R. Wang, Z. Chen, T. J. Bao, M. Montes-de-Oca, Y. Cheng, M. Lin, L. Torresani and A. T. Campbell, “CarSafe App: Alerting Drowsy and Distracted Drivers using Dual Cameras on Smartphones”, In MobiSys’13, June 25-28, 2013.

[6] Sahayadhas, K. Sundaraj, and M. Murugappan,” Drowsiness detection during different times of day using multiple features”, Australasian Physical & Engineering Sciences in Medicine, Volume 36, Issue 2, pp 243-250, June 2013.

[7] Asma-Ul-Husna., Amit Roy, Gautam Paul, MilonKanti Rah. “Fatigue Estimation through Face Monitoring and Eye Blinking”, the 3rd International Conference on Mechanical, Industrial and Energy Engineering (ICMIEE 2014), 25-26 December 2014, Khulna, Bangladesh.

[8] Calculating Geographic Distance: Concepts and Methods – Frank Ivis. [online].

[9] Eyeblink detection with OpenCV, Python, and Libby Adrian Rosebrock on April 24, 2017 [online].G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[10] Z. Arifin, M. R. Ibrahim and H. R. Hatta, "Nearest tourism site searching using Haversine method," 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), pp. 293-296, 2016.

[11] A. Forero and F. Calderon, "Vehicle and pedestrian video-tracking with classification based on deep convolutional neural networks," 2019 XXII Symposium on Image, Signal Processing and Artificial Vision (STSIVA), pp. 1-5, 2019.