DRIVER DROWSINESS DETECTION SYSTEM

SOCIALLY RELEVANT PROJECT REPORT

Submitted by

2018115047 Keerthana. S

2018115059 Murugan. A

2018115137 Yoheshwar. S

submitted to the Faculty of

INFORMATION SCIENCE AND TECHNOLOGY

for the completion of

SOCIALLY RELEVANT PROJECT

*for **THIRD YEAR***

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY

CHENNAI - 600 025

DECEMBER – 2020

ABSTRACT

SLEEPALERT

- This is an android based drowsiness detection system .
- We have developed a Smartphone based application which monitors the eyes of the driver and alerts the driver when they closes their eyes for more than 1.5 seconds.
- Front camera of the Smartphone is used to monitor the face of the user in real time.
- Drivers need to have mobile stands for vehicles to use the system in real world.

Hardware Requirements :

- Android Device with minimum sdk 21
- Mobile stand for vehicles

Software Requirements :

- Android studio

TABLE OF CONTENTS

CHAPTER – 1

INTRODUCTION

This chapter gives an overview of the process related to driver drowsiness detection system. This system elaborates the challenges that are to be considered in the development of this proposed system.

1.1 PURPOSE

- Drivers who do not take regular breaks when driving long distances run a high risk of becoming drowsy a state which they often fail to recognize early enough according to the experts.
- Studies show that around one quarter of all serious motorway accidents are attributable to sleepy drivers in need of a rest, meaning that drowsiness causes more road accidents than drink-driving.
- Attention assist can warn of inattentiveness and drowsiness in an extended speed range and notify drivers of their current state of fatigue and the driving time since the last break and makes the driver to realise his inattentiveness and make them aware of it.

1.2 DRIVER DROWSINESS DETECTION SYSTEM

- Driver drowsiness detection is a car safety technology which prevents accidents when the driver is getting drowsy.

- Various studies have suggested that around 20% of all road accidents are fatigue-related, up to 50% on certain roads.
- Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes.
- The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems.
- Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.
- Driver inattention might be the result of a lack of alertness when driving due to driver drowsiness and distraction.
- Driver distraction occurs when an object or event draws a person’s attention away from the driving task.
- Unlike driver distraction, driver drowsiness involves no triggering event but, instead, is characterized by a progressive withdrawal of attention from the road and traffic demands.
- Both driver drowsiness and distraction, however, might have the same effects, i.e., decreased driving performance, longer reaction time, and an increased risk of crash involvement.

1.3 SCOPE

- Driving is a complex task where the driver is responsible of watching the road, taking the correct decision on time and finally responding to other driver's actions and different road conditions.
- Based on Acquisition of video from the camera that is in front of driver perform real-time processing of an incoming video stream in order to infer the driver's level of fatigue if the drowsiness is Estimated then the output is send to the alarm system and alarm is activated.
- This makes the driver aware of his self inattentiveness.

1.4 SIGN OF DROWSINESS

The signs of the driver drowsiness are :

- Driver may be yawn frequently.
- Driver is unable to keep eyes open.
- The driver can't remember driving the last few miles.
- Drift into the other lane or onto the shoulder of the road.

1.5 FACTORS CAUSING DRIVER DROWSINESS

- Driver Fatigue is often caused by four main factors: sleep, work, time of day, and physical.
- Often people try to do much in a day and they lose precious sleep due to this. Often by taking caffeine or other stimulants people continue to stay awake.

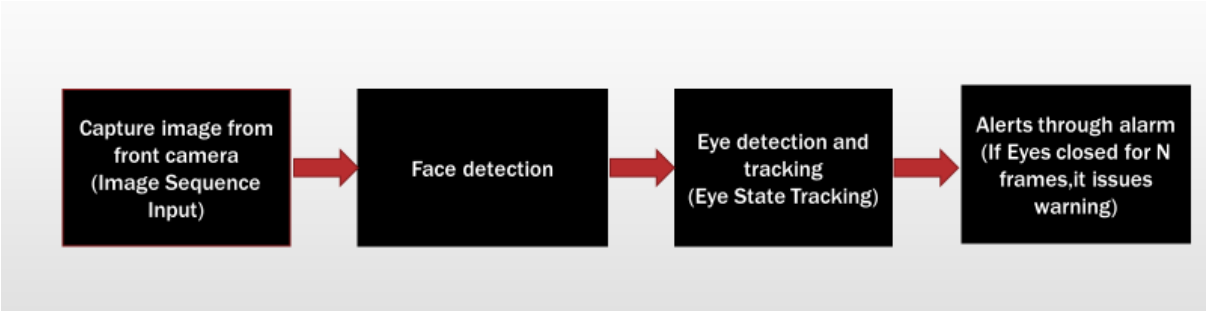
- The lack of sleep builds up over a number of days and the next thing that happens is the body finally collapses and the person falls asleep.
- Time of day factors can often affect the body. The human brain is trained to think there are times the body should be asleep. These are often associated with seeing the sunrise and sunset. Between the hours of 2 AM and 6 AM, the brain tells the body it should be asleep.
- Extending the time awake will eventually lead to the body crashing.
- The final factor is a person's physical condition.
- People sometimes are on medications that create drowsiness or have physical ailments that cause these issues.
- Being physically unfit, by being either under or overweight, will cause fatigue.
- Additionally, being emotionally stressed will cause the body to get fatigued quicker.

CHAPTER 2

SOFTWARE REQUIREMENTS AND SPECIFICATIONS

This chapter discusses about the perspective of the product and the description of each and every modules involved and the algorithms are also discussed here.

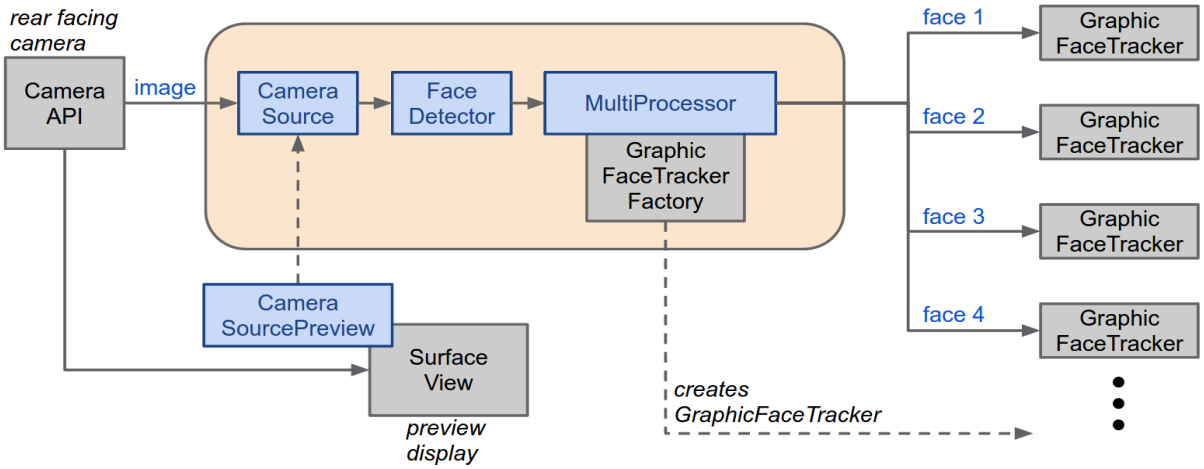
2.1 PRODUCT PERSPECTIVE



2.2 PRODUCT FUNCTION

- Object detection technology needs a specific face to be detected.
- If drivers close their eyes for a long period, the sensor which is the front camera in the smartphone will catch and process this event
- Then it triggers the system to give sound alert to the user.
- Vision Library is used in processing the detected image

2.3 face-tracker-pipeline USED IN VISION LIBRARY



2.4 CREATING THE FACE DETECTOR PIPELINE

- The code for setting up and executing face tracking is in FaceTrackerActivity.java, which is the main Activity for

this app. Typically the face detector is specified in the onCreate method.

```
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);

   // other stuff...

   FaceDetector detector = new FaceDetector.Builder()
      .build(getApplicationContext());
```

- The FaceDetector.Builder specifies the properties of the face detector and initiates its creation.

```
detector.setProcessor(
   new MultiProcessor.Builder<Face>()
      .build(new GraphicFaceTrackerFactory()));
```

- Create a camera source to capture video images from the camera, and continuously stream those images into the detector and its associated processor pipeline.

```
mCameraSource = new CameraSource.Builder()
    .setRequestedPreviewSize(640, 480)
    .setFacing(CameraSource.CAMERA_FACING_BACK)
    .setRequestedFps(30.0f)
    .build(getApplicationContext(), detector);
```

- Here, we use a camera source preview UI to display the camera video and a graphic overlay to the user. We initialize the UI and start the camera like this: *mPreview.start(mCameraSource, mGraphicOverlay);*

2.5 MULTIPROCESSOR

- The MultiProcessor is a component for working with an arbitrary number of detected items (in this case, faces). *(See **2.3** for img ref.)*
- The FaceDetector may detect multiple faces in each image frame received from the camera. Each face corresponds to a distinct face identity, as specified by the "tracking" setting seen before. The multiprocessor creates a Tracker<Face> instance for each face identity that it sees. It uses the factory GraphicFaceTrackerFactory, supplied by custom code above, to create new face tracker instances as needed:

```
private class GraphicFaceTrackerFactory
  implements MultiProcessor.Factory<Face> {
    @Override
    public Tracker<Face> create(Face face) {
      return new GraphicFaceTracker(mGraphicOverlay);
    }
  }
```

- As new faces are encountered, the multiprocessor will use this factory to create a new GraphicFaceTracker instance for each face. As those faces move over time, updates are routed to each of the appropriate face tracker instances. When a face is no longer visible, the multiprocessor will dispose of its associated face tracker instance. In this way, we dynamically create/track/destroy an individual face tracker for each face that we encounter in the app.

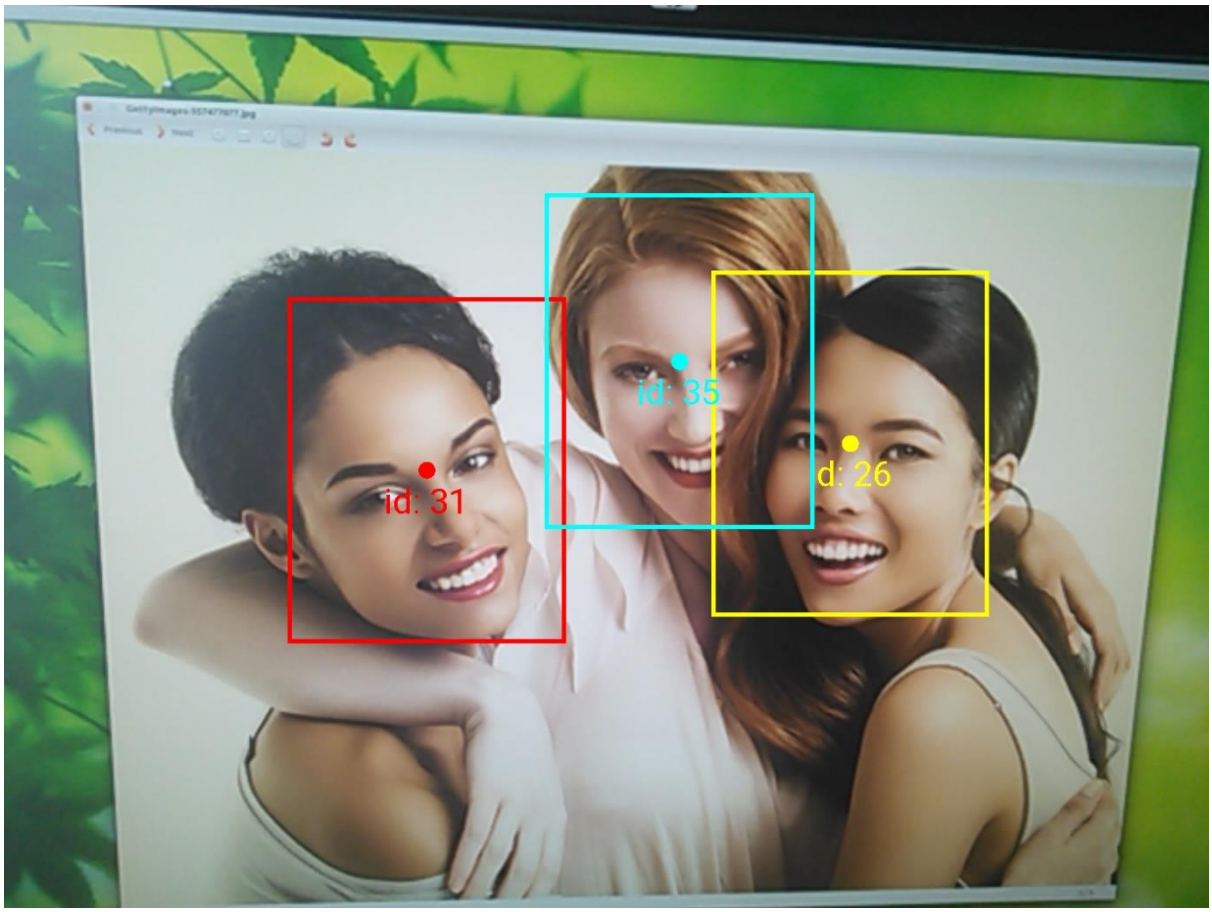
Below is the implementation of GraphicFaceTracker, which holds the state associated with an individual face:

```
private class GraphicFaceTracker extends Tracker<Face> {
```

```
    // other stuff

    @Override
    public void onNewItem(int faceId, Face face) {
        mFaceGraphic.setId(faceId);
    }

    @Override
    public void onUpdate(FaceDetector.Detections<Face> detectionResults,
                         Face face) {
        mOverlay.add(mFaceGraphic);
        mFaceGraphic.updateFace(face);
    }

    @Override
    public void onMissing(FaceDetector.Detections<Face> detectionResults)
{
        mOverlay.remove(mFaceGraphic);
    }

    @Override
    public void onDone() {
        mOverlay.remove(mFaceGraphic);
    }
}
```

- Each GraphicFaceTracker instance maintains an associated FaceGraphic instance, which is a graphics object that is added as part of the overlay of the camera video. This

instance is created initially when the face is first encountered, updated as the face changes, hidden when the face is temporarily missing, and is removed when the face is no longer visible.



2.6 FUNCTION TO DETECT THE OPEN EYE

- **public float getIsLeftEyeOpenProbability ()**
  - Returns a value between 0.0 and 1.0 giving a probability that the face's left eye is open.
  - This returns UNCOMPUTED_PROBABILITY if the probability was not computed. The probability is not computed if eye open classification is not enabled via setClassificationType(int) or the LEFT_EYE landmark was not found.

- Returns the probability for the face's left eye being open

- **public float getIsRightEyeOpenProbability ()**
  - Returns a value between 0.0 and 1.0 giving a probability that the face's right eye is open.
  - This returns UNCOMPUTED_PROBABILITY if the probability was not computed. The probability is not computed if eye open classification is not enabled via setClassificationType(int) or the RIGHT_EYE landmark was not found.
  - Returns the probability for the face's right eye being open.

2.7 EYE BLINKING BASED TECHNIQUE TO DETECT DROWSINESS

- In this eye blinking rate and eye closure duration is measured to detect driver's drowsiness.
- Because when driver felt sleepy at that time his/her eye blinking and gaze between eyelids are different from normal situations so they easily detect drowsiness".
- In this system the position of irises and eye states are monitored through time to estimate eye blinking frequency and eye close duration.
- And in this type of system uses a remotely placed camera or mobile camera to acquire video and computer vision methods are then applied to sequentially localize face, eyes and eyelids positions to measure ratio of closure.
- Using these eyes closer and blinking ration one can detect drowsiness of driver.

- Such a system, mounted in a discreet corner of the car, could monitor for any signs of the head tilting, the eyes drooping, or the mouth yawning simultaneously and it alarms so that the driver aware of his self state and stops driving ,take a nap or something and then after he becomes normal he continues driving.
- That alarm may come with the picture of his family which makes him to take a break while driving in emotionally too.

CHAPTER - 3

FUNCTIONAL REQUIREMENTS

The purpose of related work is to provide an overview by reviewing scholarly literature relevant to a topic, identify areas of study by pointing the way forward further research, analysis strength and weakness of previous research.

3.1 SYSTEM REQUIREMENTS

3.1.1 HARDWARE REQUIREMENTS

- Android Device with minimum sdk 21 for using our application.
- Mobile stand for vehicles.

3.1.2 SOFTWARE REQUIREMENTS

- **Android Studio** for programming the android application to implement it in android smart phones.

CHAPTER – 4

IMPLEMENTATION AND RESULTS

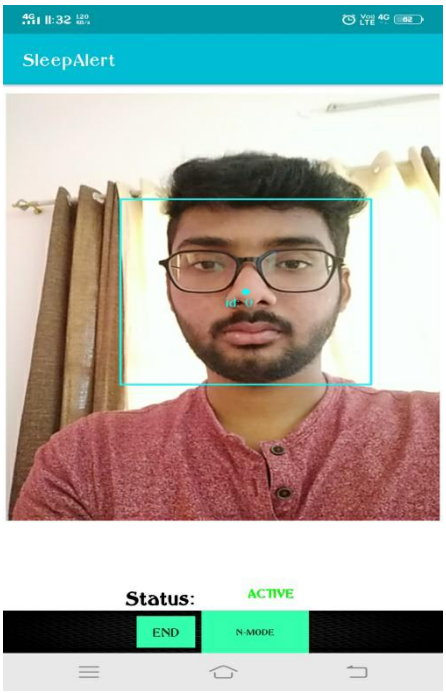
This chapter discusses the implementation of the **Sleepalert** (DRIVER DROWSINESS DETECTION APPLICATION)

MAIN PAGE:

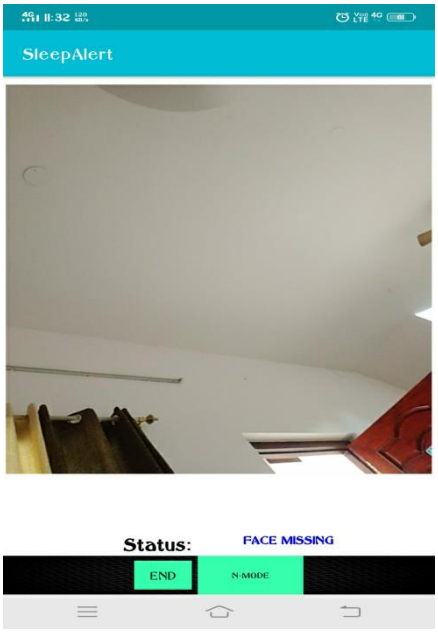
- This is the main page of our application. Here we can see a magic button.

- If we click the start button, we can start the monitoring process.

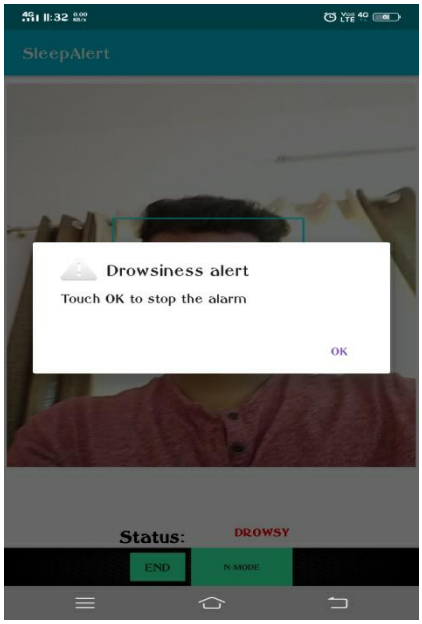
FACE TRACKER PAGE:



- Here we can see the region of interest around the face. So the status is active.

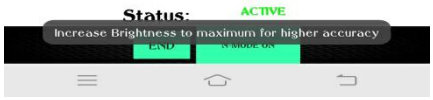


- If the face is not inside the region of interest, the status will be “FACE MISSING”.
- Let’s start the eye detection.
- For that, let the eyes be closed.

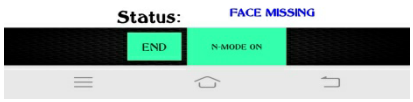


- When the eyes are closed continuously for 1.5 seconds, the alarm will be invoked. We can also see the status as “DROWSY”.

N-MODE PAGE:



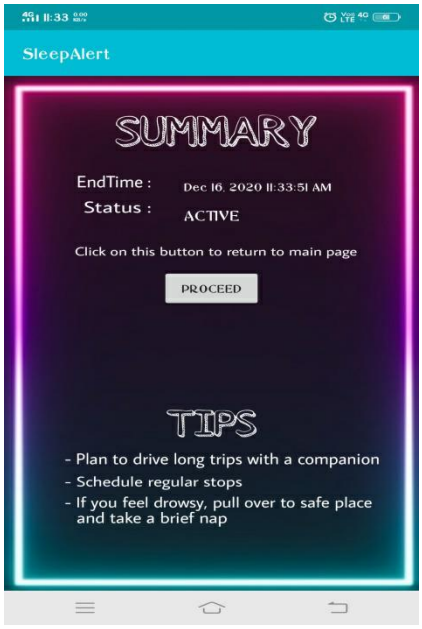
- We can also invoke Night Mode, by just clicking the N-MODE button.
- In that mode, the face won’t be visible, but it will be detected.
- We can see the status as “ACTIVE”.



- If the face is moved out of the reach of camera, the status will be “FACE MISSING”.
- Let the eyes be closed for the detection.

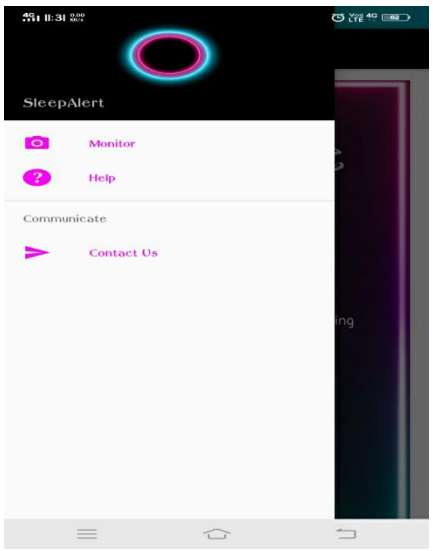
- When the eyes are closed continuously for 1.5 seconds, the alarm will be invoked. We can also see the status as “DROWSY”.
- We can also see “END” button at the bottom.

SUMMARY PAGE :



- If we click the END button, we can see a page containing the summary of the detection (i.e) End Time and status.
- We can also see some tips for the users.
- If we click the PROCEED button, it returns to the main page.

NAVIGATION WINDOW PAGE:



- In the main page, we can see a drawer.
- If we click the drawer, the navigation window opens with three options, i) monitor, ii) help, iii) contact us.
- Monitor page is nothing but the main page.

HELP PAGE :

- Help page contains instructions about how to use this application.

CONTACT US:

- Contact page contains the contact information.

CHAPTER – 5

CONCLUSION

- As described throughout the presentation, many technologies exist to detect driver fatigue.
- Currently, the number one selling product in the market is nothing more than a reed switch to detect head angle tilt. This product is extremely limited and not very effective.
- The product made by BMW and integrated into their high end cars to detect driver fatigue behaviour is slightly more effective is detection but lack proper notification to warn a driver. The current market and technologies is in its infancy mode. New technologies keep emerging using different techniques.

REFERENCES :

We referred android studio official website for syntax and understand the android studio
https://developer.android.com/reference/android/util/Log