Non RRI Project - C72

DriveX: Using AI Sampling and Machine Learning to save lives from drowsy driving

Fnu Yash [11]
Aniket Gupta [10]

INTRODUCTION

- Drowsy driving is a major cause of on-road accidents
    - 6,400 people die due to fatigued/drowsy driving every year (NSC)
    - About 325,000 car crashes reported (AAA foundation for Traffic Safety)
    - 110,000+ people injured (350% more than reported by police)
    - 50% of cases involve persons under the age of 25
- DriveX: An app used to alert drowsy drivers on the road
    - Camera based app/tool that continuously monitors driver's eyes
    - Calculates eye's squint using on AI sampling and stores in a database for easier access next time
    - Multiple users per app/tool supported
    - Alarma driver when eye ratio is below the custom threshold value specified by the user

TOOL

- Used OpenCV to scan users' faces through a computer's camera.

```
C:\Users\Fnu Yash>pip install opencv-python
```

www.blogspot.com

- Used cv2's grayscale feature to convert face image to gray image for greater accuracy.

```
grayImage = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

- DLIB to landmark face (tool file downloaded from reddit, link in references).
  - DLIB is an AI from Davis E.King. They have used thousands of human faces to teach the AI how to look for contours and decipher the human face. Due to this, DLIB is only able to provide us with 68 landmarks. While we could have used Google's Cloud Vision API or Nvidia's AI, we opted for DLIB because this is less resource intensive, and has everything we need, meaning this is the most efficient for us.

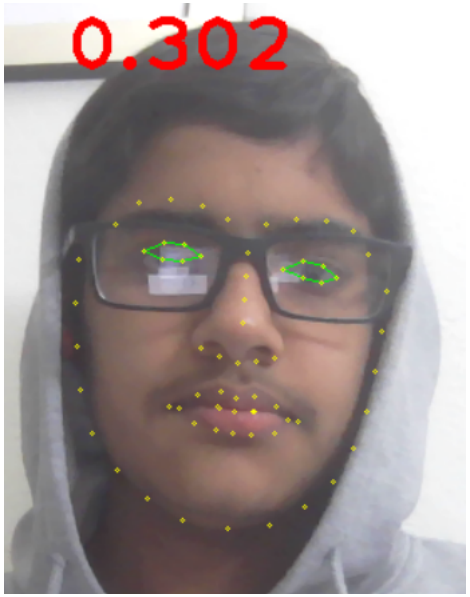
(Source: pyimagesearch.com)

```
detector = dlib.get_frontal_face_detector()
dlib_facelandmark = dlib.shape_predictor("C:/Users/Fnu Yash/Desktop/Python/Face_Detection/shape_predictor_68_face_landmarks.dat")
```

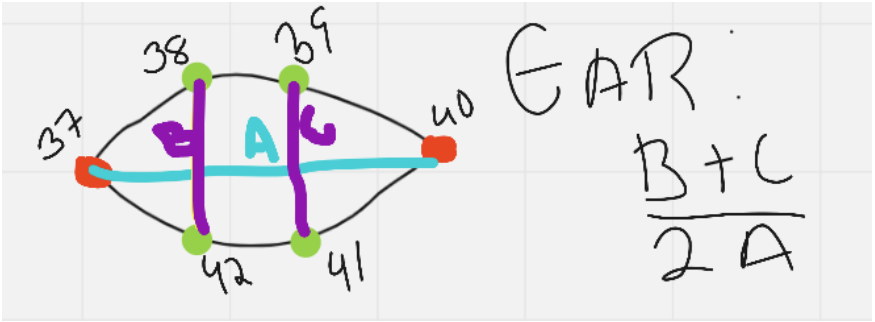
DEVELOPMENT STEPS

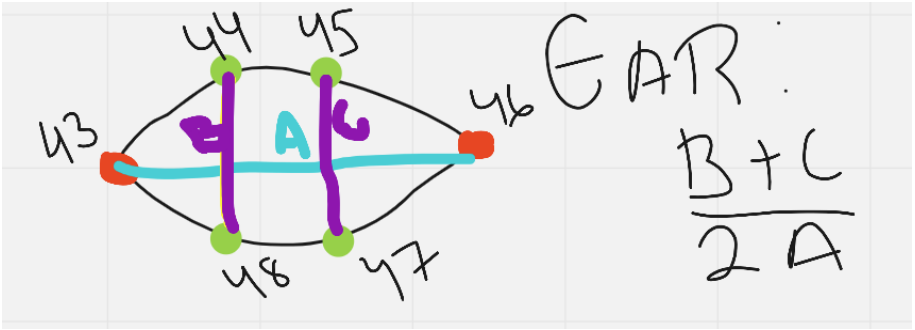
1. Initializing and capturing data from camera using opencv

```
while True:
    _, frame = cap.read()
    grayImage = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow("Face Landmarks", frame)
    cv2.waitKey(1)
```



2. Face detector using DLIB
    a. Load shape predictor file from reddit
    b. Landmark the face
    c. Calculate distance from 37-40 and average with distance from 38-42 and 39-41 (left eye) and 43-46, 44-48, and 45-47 (right eye)
        i. Averaging these two EARs gives us the final EAR





DEVELOPMENT STEPS (cont.)

earInfo.txt

```
yash: 0.26
aniket: 0.25
testdummy1: 0.5
testdummy2: 0
testdummy3: 0.28
```

3. User EAR retention system
    a. Format file
        i. [NAME]: [EAR]
    b. When given name, split at [:] and scan for name
        i. When found user part 2 of split (EAR)
    c. If the name does not exist, ask the user to enter EAR, and store and format it.

```
file = open("C:/Users/Fnu Yash/Desktop/Python/DriveX/earInfo.txt", 'r')
for line in file:
    splitLine = line.split()
    if splitLine[0] == (name+":"):
        print("Your information has been found!")
        EAR = float(splitLine[1])
file.close()
```

```
if (ratioFinal < float(EAR) and not startCounter):
    startCounter = True
    t0 = time.time()

if startCounter:
    if (time.time() - t0 >= 3) and not alreadyPlaying:
        mixer.music.play()
        drowsy = True
        alreadyPlaying = True

if (ratioFinal >= float(EAR) and startCounter):
    mixer.music.stop()
    alreadyPlaying = False
    startCounter = False
    drowsy = False
```

4. Alarm System
    a. When EAR dips, start timer
        i. If already running, reset timer
    b. If EAR returns to control and timer < time specified: pause and reset the timer.
        i. If timer > time: play alarm until EAR returns to above threshold.

```
mixer.init()
mixer.music.load('C:/Users/Fnu Yash/Desktop/Python/DriveX/alarm_tone.mp3')
alreadyPlaying = False
startCounter = False
drowsy = False
bot = True
```

RESULTS/DISCUSSION

- Procedure to use app:
  1. Run app from run button
  2. Enter your name
  3. Enter input method (if new user)
     a. If input method = 1, navigate to given path to calculate EAR
        i. Re-run main file, input method = 2, and then enter your EAR
- Test Trial #1
  - When the user entered their name, the app was not checking if the user already existed.
    - FIX: In line 38, [splitLine] was being compared as a array instead of a string. Missing ([0]) to make [splitLine[0] = line.split()]

```
35  file = open("C:/Users/Fnu Yash/Desktop/Python/DriveX/earInfo.txt", 'r')
36  for line in file:
37      splitLine = line.split()
38      if splitLine == (name+":"):
39          print("Your information has been found!")
40          EAR = float(splitLine[1])
41  file.close()
```

- Test Trial #2
  - Program crashed when we tried to close it. Camera screen would just relaunch and would not stop until the process was manually shut down.

RESULTS/DISCUSSION (cont.)

- FIX: Added a keyboard listener that looks for a key press and assigned it to sys.exit() which ends the program.

```
if keyboard.is_pressed('q'):
    break
```

- Test Trial #3
  - Program was case sensitive and would decipher user names “Bob”, “BOB”, “bob”, “bOb”, and “boB” as five different people.
    - FIX: We converted user input to lowercase letters before performing any operations.

```
name = name.lower()
```

- Test Trial #4
  - After implementing the original timer code shown, the program would pause, freeze, and sometimes crash. (Original code →)
    - FIX: We scrapped the code and remade the timer by implementing it in the main while loop so that loop wouldn’t pause or freeze trying to run another while loop. (New code ↓)

```
def timer():
    start = time.time()
    time.process_time()
    iteration = 0
    shouldRepeat = True
    alreadyPlaying = False
   # print(seconds) #testing
    print(start) #testing

    while shouldRepeat:
        iteration = time.time() - start
        print(time.time()) #testing
        print(iteration) #testing
        time.sleep(1)
        if iteration >= 3:
            if alreadyPlaying == False:
                mixer.music.play()
                alreadyPlaying = True
            if(main.ratioFinal>0.4):
                mixer.music.stop()
                shouldRepeat = False


while True:
    if(main.ratioFinal < 0.4): timer()
```

(The drowsy variable is to print DROWSY on the screen for the user.)

```
if drowsy:
    cv2.putText(frame, "Drowsy", (250, 100),
                cv2.FONT_HERSHEY_PLAIN, 3, (0, 0, 255), 4)
```

```
if (ratioFinal < float(EAR) and not startCounter):
    startCounter = True
    t0 = time.time()

if startCounter:
    if (time.time() - t0 >= 3) and not alreadyPlaying:
        mixer.music.play()
        drowsy = True
        alreadyPlaying = True

if (ratioFinal >= float(EAR) and startCounter):
    mixer.music.stop()
    alreadyPlaying = False
    startCounter = False
    drowsy = False
```

RESULTS/DISCUSSION (cont.)

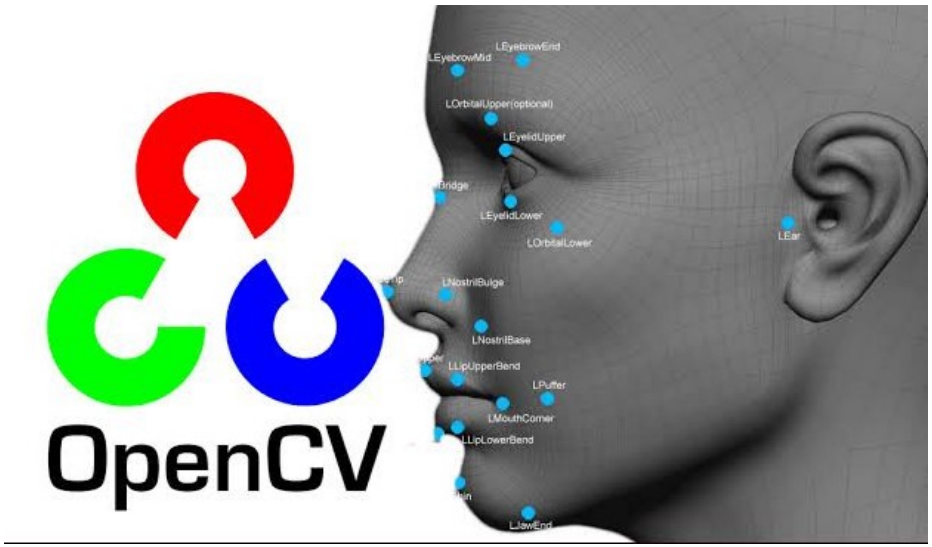
Source: tntrafficsafety.org/drowsy-driving

There are 8 usual signs of a driver being drowsy: drifting out of your lane, inability to keep your head up, missing your turns/exits, feeling restless, failure to remember past, recent things (like landmarks, distance traveled), and one of the main ones, having trouble focusing and keeping your eyes open. From our results, we now know that our program is capable of identifying one of the main signs of drowsiness to better ensure the safety of drivers as well as passengers and other people in cars and/or the sidewalks.

RESULTS/DISCUSSION (cont.)

Our prototype is an improvement of current safety features implemented in cars and other technologies. There are softwares made by companies to analyze the car's surroundings (ex: Tesla's Autopilot). It is able to identify if the car is properly still within the lane, any incoming cars, pedestrians, bicycles, before making a turn and/or lane switch. There are plenty of others but all of these are safety measures based off of the vehicle. Our prototype is a safety measure based on the person, the driver, which also makes our software unique considering there hasn't been much on analyzing the driver instead of the surroundings. Considering outside forces is one good way to decrease chances of accidents, but in many disasters/accidents, the main cause is the driver. If we can fix the problem from its root, there is a much higher chance where accidents can be prevented.

Source: www.medium.com



CONCLUSION

After further testing and fixing the bugs in our trial, the program works perfectly as intended. It was able to meet our engineering goal. It is able to ask the user for his/her name and check if the program has an EAR value stored for that user, if not, will calculate an EAR for them, and then properly identify the user's face and check if the user is drowsy from their eyes closing.

- There are also many other future applications that can be built on top of DriveX
    - Integrate with other mapping apps and use phone camera to for easy access
    - Integrate with car's internal software (along with sensors to make the app more effective)
    - Broadcast driver state with nearby drivers (car to car communication) , family members, and/or law enforcement authorities.
    - Analyze short-term and long-term driver behaviors and driving patterns to provide more tailored experiences.

REFERENCES

- https://www.nsc.org/road-safety/safety-topics/fatigued-driving
- DLIB Documentation
  - http://dlib.net/
- OpenCV Documentation
  - https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
- Downloads
  - https://www.python.org/downloads/
  - In terminal
    - pip install opencv-python
    - pip install cmake
    - pip install dlib
    - pip install pygame
  - Alarm
    - https://www.zapsplat.com/sound-effect-category/emergency/
  - Shape Predictor tool for DLIB
    - https://github.com/davisking/dlib-models/blob/master/shape_predictor_68_face_landmarks.dat.bz2