***Project2 :DROWSINESS DETECTION***

Driver drowsiness detection is a [car safety](https://en.wikipedia.org/wiki/Car_safety) technology which helps prevent accidents caused by the driver getting drowsy. Warnings can be [visual](https://en.wikipedia.org/wiki/Visual), audible, [vibrating](https://en.wikipedia.org/wiki/Vibrating), or tactile. Here we use a camera for computer vision to detect the drowsiness.

***------------------------------------------------------------------------------------------------------------------------***

**Step 1**

Import the required libraries of python OpenCV to the script for computer vision algorithms.

Here we use opencv 3.4.1 in our scripts.

*import cv2*

***------------------------------------------------------------------------------------------------------------------------***

**Step 2**

` Capture the frames for image processing in one of the method given below.

1. *cap =* [cv2.VideoCapture](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_video_display/py_video_display.html)*('test/video4.mp4') --> from video file*

*2.cap = cv2.VideoCapture(0) --> from camera*

*Note : Here ‘0’ indicates the primary cmera of your computer .We can choose the desired camera by its numer in here.*

***------------------------------------------------------------------------------------------------------------------------***

**Step 3**

Now we create the cascade and initialize it with our face cascade. This loads the face cascade into memory so it’s ready for use. Remember, the cascade is just an XML file that contains the data to detect faces.

*face\_cascade =* [*cv2.CascadeClassifier*](https://docs.opencv.org/3.0-beta/modules/objdetect/doc/cascade_classification.html)*('haarcascade\_frontalface\_alt.xml')*

we also load eye csacade in similar way.These cascade files can be found in OpenCV build folder.

*eye\_cascade =* [*cv2.CascadeClassifier*](https://docs.opencv.org/3.0-beta/modules/objdetect/doc/cascade_classification.html)*('haarcascade\_eye.xml')*

***------------------------------------------------------------------------------------------------------------------------***

**Step 4**

Here we read the image and convert it to grayscale. Many operations in OpenCV are done in grayscale.

*ret, frame =* [*cap.read*](https://docs.opencv.org/3.0-beta/modules/videoio/doc/reading_and_writing_video.html)*()*

1.ret : True-> for sucessfull capture

False- >for faled capture

2.frame:The image frame

*frame = [cv2.resize](https://docs.opencv.org/3.0-beta/modules/imgproc/doc/geometric_transformations.html" \l "resize)(frame, None, fx=ds\_factor, fy=ds\_factor, interpolation=cv2.INTER\_AREA)*

1.fx,fy : scale factor on x axis and y axis respectively.

**Step 5**

Detector required grayscale imgae,So we do a colour conversion from RG to grayscale.

*gray = [cv2.cvtColor](https://docs.opencv.org/3.0-beta/modules/imgproc/doc/miscellaneous_transformations.html" \l "cvtcolor)(frame, cv2.COLOR\_BGR2GRAY)*

cv2.cvtColor is an OpenCV function to convert images to different color spaces. It takes as input an image to transform, and a color space code (like cv2.COLOR\_BGR2RGB) and returns the processed image.

***------------------------------------------------------------------------------------------------------------------------***

**Step 6**

Detection of faces is done using detect MultiScale OpenCV function.

*faces = face\_cascade.*[*detectMultiScale*](https://docs.opencv.org/3.0-beta/modules/objdetect/doc/cascade_classification.html)*(gray, 1.1, 5)*

detectMultiScale(image, scaleFactor, minNeighbors): This is a general function to detect objects, in this case, it'll detect cars since we called in the car cascade. If it finds a car, it returns a list of positions of said car in the form “Rect(x,y,w,h).”, if not, then returns “None”.

* Image: The first input is the grayscale image. So make sure the image is in grayscale.
* ScaleFactor: This function compensates a false perception in size that occurs when one car appears to be bigger than the other simply because it is closer to the camera.
* MinNeighbors: This is a detection algorithm that uses a moving window to detect objects, it does so by defining how many objects are found near the current one before it can declare the car found.

***------------------------------------------------------------------------------------------------------------------------***

**Step 7**

**"Detection of face and eyes using opencv cascade detection "**

1.Draw a rectangle box in each detected faces.

*for (x,y,w,h) in faces:*

[*cv2.rectangle*](https://docs.opencv.org/3.0-beta/modules/imgproc/doc/drawing_functions.html)*(frame,(x,y),(x+w,y+h),(0,0,255),2)*

* Frame:Input image
* (x,y) : Vertex of the rectangle.
* (x+w,y+h) : Vertex of the rectangle opposite to pt1 .
* (0,0,255) :Rectangle color or brightness (grayscale image).
* **2:**Thickness of lines that make up the rectangle. Negative values, like CV\_FILLED , mean that the function has to draw a filled rectangle.

2.Get the face part from the full frame.

roi\_gray = gray[y:y+h,x:x+w] # region of interest

3.Detection of eyes is done using detect MultiScale OpenCV function.

eyes = eye\_cascade.detectMultiScale(roi\_gray,2.1,7) # 2.1=scaling factor,7=minimum neighbours

4.Draw a rectangle box in each detected eyes.

for (x\_e,y\_e,w\_e,h\_e) in eyes:

cv2.rectangle(roi\_color,(x\_e,y\_e),(x\_e+w\_e,y\_e+h\_e),(0,255,0),2) # (0,255,0)=green color, 2=thickness

5.Check for eyes closed more than 5 frames

count += len(eyes)

iters += 1

if iters ==5:

iters = 0

if count == 0:

print "Drowsiness Detected!!!"

count = 0

***------------------------------------------------------------------------------------------------------------------------***

**Step 8**

Next is to display the ouput frame.

*cv2.*[*imshow*](https://docs.opencv.org/3.0-beta/doc/tutorials/introduction/display_image/display_image.html)*('output', frame)*

This is a cv2 function used to display the image. It also takes two arguments: the first one is the name of the window that will pop-up to show the picture and the second one is the image you want to display.

***------------------------------------------------------------------------------------------------------------------------*Step 9**

To escape from while loop we use a key interrupt,Once the ‘esc’ is pressed the the loop get break.ASCII value of esc is 27 in decimal and OpenCV wait for a cycle for keypress.

*if* [*cv2.waitKey*](https://docs.opencv.org/3.0-beta/doc/user_guide/ug_highgui.html)*(1) == 27:*

*break*

***------------------------------------------------------------------------------------------------------------------------***

**Step 10**

De-allocate any associated memory or device usage.

[*cap.release*](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_video_display/py_video_display.html)*()* ->This will relase the camera allocated for our project.

cv2.[destroyAllWindows](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_video_display/py_video_display.html)() -*>*This simply destroys all the windows we created using cv2.imshow(window\_name, image)

***------------------------------------------------------------------------------------------------------------------------***