***Project3 :*** ***LANE KEEP ASSIST***

In road-transport terminology, a lane departure warning system is a mechanism designed to warn the driver when the vehicle begins to move out of its lane. 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 track lanes on road.

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

**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*

*import numpy as np*

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

**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**

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

*ret, img =* [*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

**Step 5**

"**We do a colour conversion from RGB 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 edges using opencv canny edge detection."**

*edge =* [*cv2.Canny*](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_canny/py_canny.html)*(gray,1,150,3)#Canny edge detection*

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm and we will go through each stages. we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges

* gray: input image in grayscale.
* 1: Min Value.
* 150: Max value.
* **3**: Third argument is aperture\_size. It is the size of Sobel kernel used for find image gradients. By default it is 3

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

**Step 7**

**"Detection of lanes using opencv hough line detection "**

lines = cv2.[HoughLinesP](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html)(edge,1,np.pi/180,100,1,10)

Hough Transform is a popular technique to detect any shape, if you can represent that shape in mathematical form. It can detect the shape even if it is broken or distorted a little bit. First parameter, Input image should be a binary image, so apply threshold or use canny edge detection before finding applying hough transform. Second and third parameters are \rho and \theta accuracies respectively. Fourth argument is the threshold, which means minimum vote it should get for it to be considered as a line. Remember, number of votes depend upon number of points on the line. So it represents the minimum length of line that should be detected.

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

**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)

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