# -> function

Import OpenCV library to import the image manipulation functions.

cap is the variable which holds the video that is captured with the webcam or video file or live stream. And here the input video is given as test1. The video is stored in the cap variable with the help of the method VideoCapture(). We capture the every frame or image in the video using read() into the variable frame, every frame in the video would be read until the video that given as input is open.

canny\_image is the variable which is given as input to the canny edge algorithm which is used to detect the edges in the video.

**#canny():**

Firstly it checks if there is an image or not if there is no image it destroys all the windows that were created and release the cap and exits.

Later if we find an image firstly we convert the colors blue, green, red to gray using the function #cvtColor() and is saved into the variable gray.

Kernel is provided in order to stop the noise disturbances and is set to 5.

As of our knowledge we can say that the lanes are present in the lower half of the video and we crop the part in which the lanes are present using the region\_of \_interest() into which the canny\_image is sent as the parameter and that cropped image is stored in the variable cropped\_canny.

**#region\_of\_interest():**

It takes the canny\_image and obtains the height and width of the cropped image. WE create a mask to the capture the lane which our main result. The masked\_image is the variable which stores the part of the road in between the lanes and including the lanes in the form of a triangle using fillPoly() which uses the triangle with provided dimensions and fits the image into that triangle and blanks out the other parts other than the road between the lanes and the lanes.

**#houghLines():**

It gives the approximate the presence of the lanes from the image. We used the default HoughLines() and reshaped according to our convenience of the result video.

**#average\_slope\_intercept():**

It is used to get the exact location of the lane by taking the slopes of them, it takes the input and identify the lines which are important and which are not. After the it uses the make\_points() and gets the slope of those required lines.

**#make\_points():**

this function defines the slope of the line based on the image sizes and assumes the size of the line and finds the slope of the image with the size given.

**#display\_lines():**

It is used to extract the lines using the x, y coordinates and generate the red-colored line and display the that on the black-colored lines (lanes).

**#addWeighted():**

we used the default addWeighted() which increase the density or the colored line which is displayed on the actual black lines which are nothing but the lanes in the frames of the video. And it merges the both the lines so that the lanes would be highlighted without any deviations.