**Procedure and Explanation:**

1>First I input an image

img = imread("test3.png");



2>Perform lane detection on the image and store the final image into dst

>lane detection already explained in the previous given task:

cvtColor(img, greyscale\_img, COLOR\_BGR2GRAY);

blur(greyscale\_img, greyscale\_\_blur\_img, Size(3, 3));

Canny(greyscale\_\_blur\_img, greyscale\_\_blur\_img, 100, 300, 3);

dst = Scalar::all(0);

img.copyTo(dst, greyscale\_\_blur\_img);

(dst.png)

3>Process for edge detection begins:

First to make sure that the image is greyscale we convert the image into greyscale ,as edge detection is only possible on greyscale image.

Mat inputImage = dst;

if (inputImage.channels() > 1)

{

cv::cvtColor(inputImage, inputImage, CV\_RGB2GRAY);

}

**4>**Here we define the Region of an interest in an image .It is basically the region in the whole picture where we have to detect the lanes.I chose a ROI as a triangle in front of the camera of a car to detect the lanes.Hence the images which are taken from the perspective of a car’s front dash cam perform better in the test.

>For defining ROI.First I declare the co-ordinates;

int **x0 = img.cols - 1;**

int y0 = img.rows - 1;

int x1 = img.cols / 2-1;

int y1 = img.rows / 2-1;

int x2 = 0;

**int y2 = img.rows – 1;**

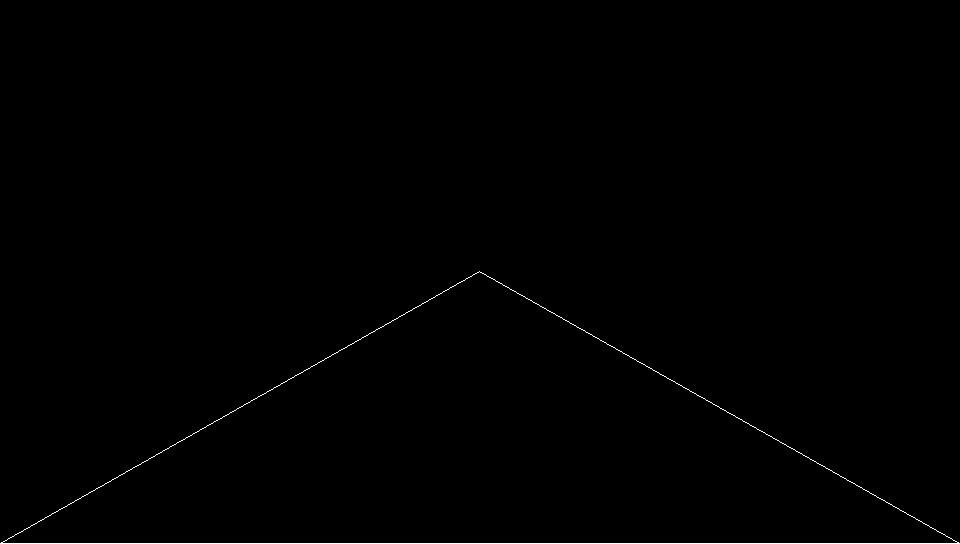
Then I make these lines on a black background and store in it LineMask image

cv::Mat lineMask = cv::Mat::zeros(inputImage.size(), inputImage.type());

cv::line(lineMask, cv::Point(x0, y0), cv::Point(x1, y1), cv::Scalar(255, 255, 0), 1, 8, 0);

cv::line(lineMask, cv::Point(x0, y0), cv::Point(x2, y2), cv::Scalar(255, 255, 0), 1, 8, 0);

cv::line(lineMask, cv::Point(x1, y1), cv::Point(x2, y2), cv::Scalar(255, 255, 0), 1, 8, 0);

(lineMask.jpg)

5>Now ,In order to Crop the image in this area,we will have to create a binary mask.

To create a binary mask,we have to perform contour detection on the image. This basically identifies the shape present in the image.It only function precisely in a black and white image ,hence linemask was made by white lines on a black background;

vector<vector<Point>> contours;

vector<cv::Vec4i> hierarchy;

cv::findContours(lineMask, contours, hierarchy, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, cv::Point(0, 0));

>After this test we perform a point polygon test ,which basically tells us which points lie inside the detected object and which points lie outside.We capture all this data in a matrix called raw\_dist;

cv::Mat raw\_dist(lineMask.size(), CV\_32FC1);

for (int i = 0; i < lineMask.rows; i++)

{

for (int j = 0; j < lineMask.cols; j++)

{

raw\_dist.at<float>(i, j) = cv::pointPolygonTest(contours[0], cv::Point2f(j, i), true);

//cout << raw\_dist.at<float>(i, j) << endl;

}

}

>Using the data stored in the raw\_dist matrix,we create the required binary matrix(mask);

cv::Mat mask = cv::Mat::zeros(inputImage.size(), CV\_8UC1);

for (int i = 0; i < mask.rows; i++)

{

for (int j = 0; j < mask.cols; j++)

{

if (raw\_dist.at<float>(i, j) < 0)//point outside the mask

{

mask.at<uchar>(i, j) = static\_cast<uchar>(0);

continue;

}

mask.at<uchar>(i, j) = static\_cast<uchar>(255);//point inside the mask

}

}



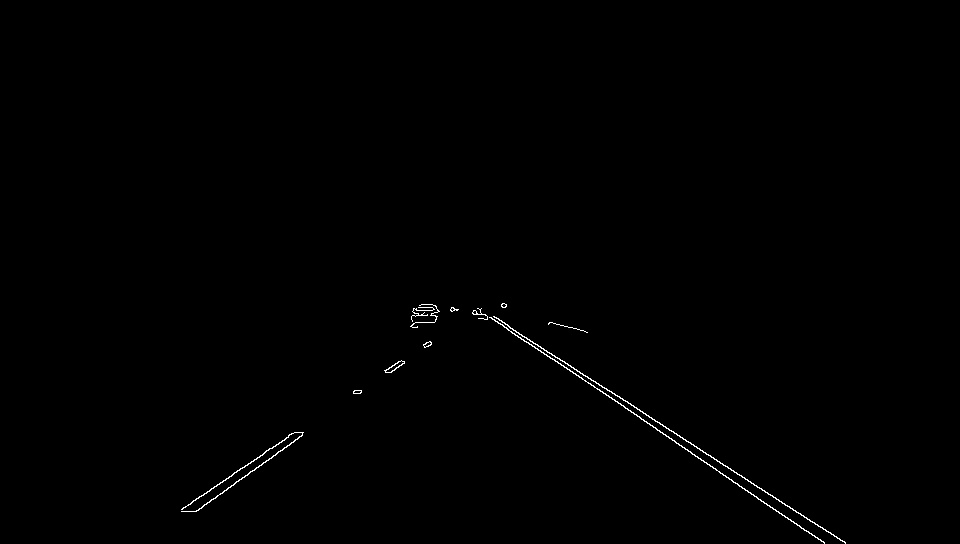
(mask.jpg)

6>We now crop the required region of interest using the above mask from the image and store it into output image;

cv::Mat invInput = inputImage;

cv::Mat outputImage;

invInput.copyTo(outputImage, mask);



(outputImage.jpg)

7>Now we use hough transform to detect any straight lines present in the image and store the end points of the lines in a vector

vector<Vec4i> linesP; // will hold the results of the detection

HoughLinesP(outputImage, linesP, 1, CV\_PI / 180, 20, 20, 30);// runs the actual detection

8>Now I have used the endpoints to calculate slope to avoid a line that is too horizontal to get printed

Now display all the detected line on the initial image

double slope\_thresh = 0.3;

for (size\_t i = 0; i < linesP.size(); i++)

{

Vec4i l = linesP[i];

double slope = (static\_cast<double>(l[3]) - static\_cast<double>(l[1])) / (static\_cast<double>(l[2]) - static\_cast<double>(l[0]) + 0.00001);

if (std::abs(slope) > slope\_thresh)

line(img, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0, 0, 255), 3, LINE\_AA);

}



(test3\_output.jpg)