Propose a solution (preferably a one-page document) mentioning the approach to automate.

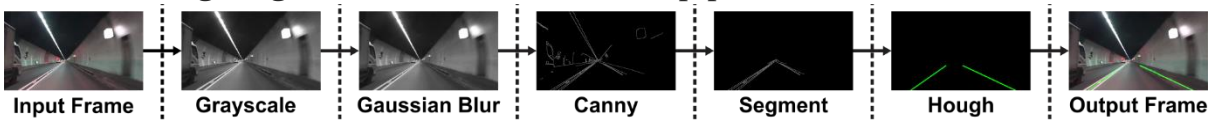
Lane detection is one of the major problem in the area of computer vision and specifically a self driving cars where the lanes will be determined and then it will be feed into the system for steering the wheels.

There are many approaches for automatically detecting the lanes from the images taken by the camera or dash cam.

1) **One of the easiest approach is to apply the classical computer vision approach.**
2) **Determining the lanes using the deep learning.**

1) Lane detection using the classical computer vision(Hough Transform)

Most lanes are designed to be relatively straightforward not only as to encourage orderliness but also to make it easier for human drivers to steer vehicles with consistent speed. Therefore, our intuitive approach may be to first detect prominent straight lines in the camera feed through edge detection and feature extraction techniques. We will be using OpenCV, an open source library of computer vision algorithms, for implementation. The following diagram is an overview of our pipeline.



In this approach we process the images taken from the camera.
Steps for detecting lanes using Hough Transformation approach.

1) **Read the image**
   The image can be read using various image processing library but majorly opencv is being utilized.
2) **Apply Canny Edge Detector**
   The Canny Detector is a multi-stage algorithm optimized for fast real-time edge detection. The fundamental goal of the algorithm is to detect sharp changes in luminosity (large gradients), such as a shift from white to black, and defines them as edges, given a set of thresholds. The Canny algorithm has four main stages:

A)Noise Reduction:-As with all edge detection algorithms, noise is a crucial issue that often leads to false detection. A 5x5 Gaussian filter is applied to convolve (smooth) the image to lower the detector's sensitivity to noise. This is done by using a kernel (in this case, a 5x5 kernel) of normally distributed numbers to run across the entire image, setting each pixel value equal to the weighted average of its neighboring pixels.

B) Intensity Gradient:-The smoothened image is then applied with a Sobel, Roberts, or Prewitt kernel (Sobel is used in OpenCV) along the x-axis and y-axis to detect whether the edges are horizontal, vertical, or diagonal.

C) Non-maximum suppression is applied to "thin" and effectively sharpen the edges. For each pixel, the value is checked if it is a local maximum in the direction of the gradient calculated previously.

D) Hysteresis thresholding:-After non-maximum suppression, strong pixels are confirmed to be in the final map of edges. However, weak pixels should be further analyzed to determine whether it constitutes as edge or noise. Applying two pre-defined minVal and maxVal threshold values, we set that any pixel with intensity gradient higher than maxVal are edges and any pixel with intensity gradient lower than minVal are not edges and discarded. Pixels with intensity gradient in between minVal and maxVal are only considered edges if they are connected to a pixel with intensity gradient above maxVal.

3) Applying the triangular mask for segmenting the lane area.
4) Hough Transform

2)Lane Detection using CNN(Semantic Segmentation)

In the code I have used the Semantic Segmentation Approach to segment the lane lines from the given figure.

The primary goal of a segmentation task is to output pixel-level output masks in which regions belonging to certain categories are assigned the same distinct pixel value. If you color-code these segmentation masks by assigning a different

color for every category for visualizing them, then you'll get something like an image from a coloring book for kids. Segmentation has been in the domain of computer vision and image processing since very long and it is very useful in detecting and segmenting the image of interest among all the other objects present in the image.

In the task I have used the DeepLabv3 by Google for the segmentation task(based on the given dataset where the mask and the ground truth images are given as the dataset). Since the dataset is very limited for the task(500 images for the task transfer learning would be a good approach for segmenting the image here lane lines from the images).

Procedure of Transfer Learning:-

1. We change the target segmentation sub-network as per our own requirements and then either train a part of the network or the entire network.
2. The learning rate chosen is lower than in the case of normal training. This is because the network already has good weights for the source task. We don't want to change the weights too much too fast.
3. Also sometimes the initial layers can be kept frozen since it is argued that these layers extract general features and can be potentially used without any changes.

Here I have used the DeepLabV3 Model by Google. Torchvision as pre-trained model and in the code the pre-trained model is used and fine tuned as per the dataset.Codes are available in the repo.

The major approach or step is as follows:-

First, we get the pre-trained model using the models.segmentation.deeplabv3_resnet101 method that downloads the pre-trained model into our system cache. Note resnet101 is the backbone for the

deeplabv3 model obtained from this particular method. This decides the feature vector length that is passed onto the classifier.

The second step is the major step of modifying the segmentation head i.e. the classifier. This classifier is the part of the network and is responsible for creating the final segmentation output. The change is done by replacing the classifier module of the model with a new DeepLabHead with a new number of output channels. 2048 is the feature vector size from the resnet101 backbone. If you decide to use another backbone, please change this value accordingly.

Finally, we set the model is set to train mode. This step is optional.

The model is trained for 25 epochs with a learning rate of 0.0001 and Adadelta optimizer.