Lane Detection

Udacity Term - 1

Period October to January

1. System & Software Specification

OS - Windows 7
Hardware: Intel i7 core CPU
Programming language: Python 3.x
Python Libraries used:

- OpenCV: library name "cv2" . Used for image processing
- Numpy: Array related functionality
- Matplotlib: used for plotting images
- Math: Used for finding square roots

2. Description

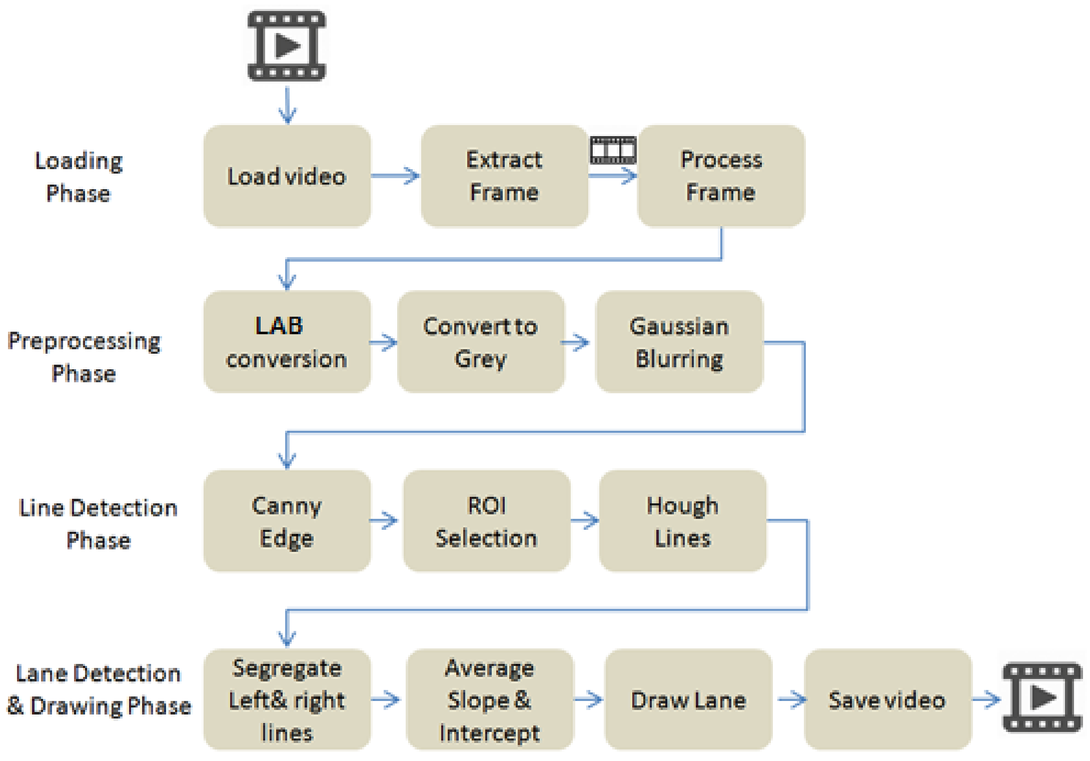
Objective: Goal is the detect the lanes in the below entities, and mark the same

- Test Images
- SolidWhiteRight.mp4
- SolidYellowLeft.mp4
- Optional: challenge.mp4

A program has to be implemented in python, which can load the given test data, either image or video. Image processing techniques to be adopted to detect the lanes in the same and marked. The program has to work for all the frames in the loaded video. The output, must be saved with the markings on the lane.

3. Approach

The overall flow of the program is as shown below



3.1. Loading Phase

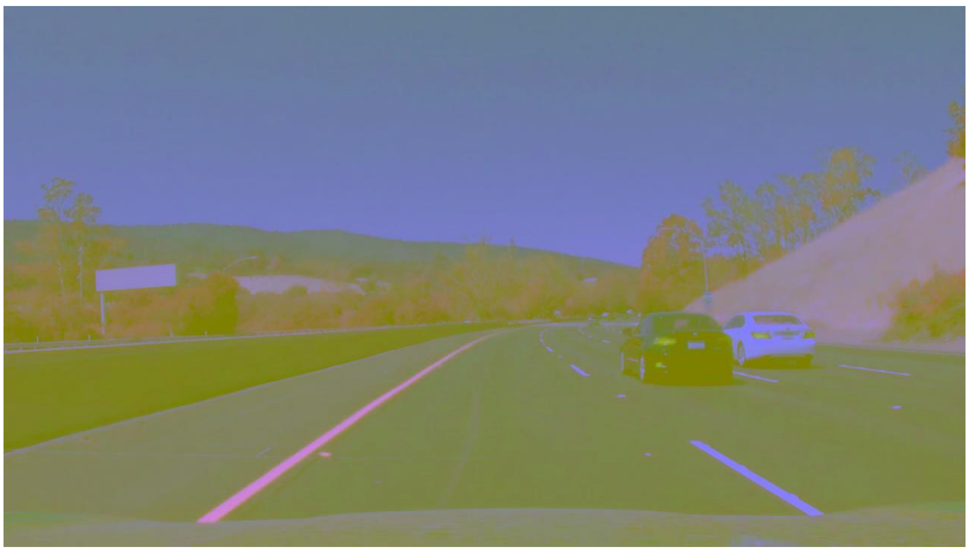
The Video files are loaded using the opencv python library module “VideoCapture” . All the frames are passed to a function “processFrame” for processing.

3.2. Preprocessing Phase

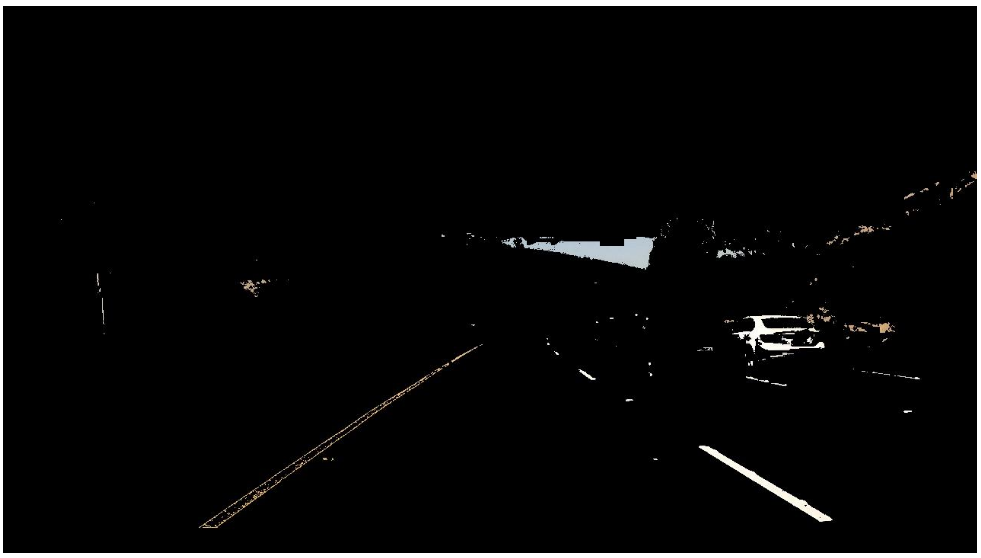
The first part of the flow is to do some preprocessing, so that, extracting lanes becomes easier. The preprocessing phase consists of three stages.

Conversion to LAB scale

In this stage, we convert the loaded frame to LAB space. An output , after conversion, is as shown below.



Converting to LAB space helps us to easily identify things belonging to certain color. To do this, we need to set the lower and upper LAB value of the color we are interested in.
In the current video, we are interested in white and yellow color, so we set the lower and upper LAB values for these two colors. We convert the given frame to LAB space, and then extract the masks based on the upper and lower thresholds.



Conversion to grey scale
The LAB converted image is converted to grey scale. An output is as shown below

Applying Gaussian blurring
Gaussian blurring is applied to smoothen the image. An output is as shown below. A kernel size of 5 is used.

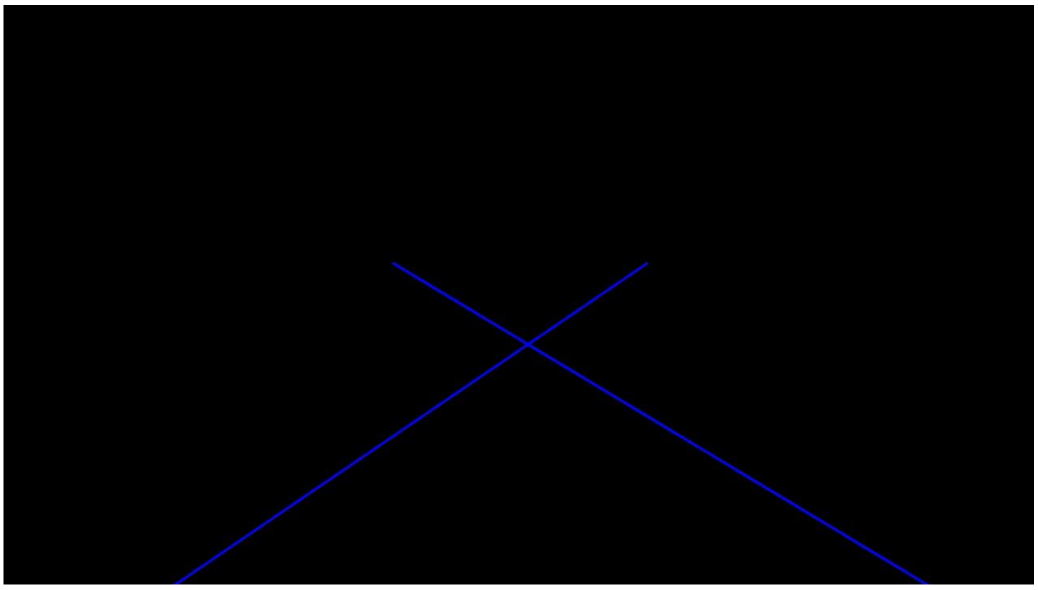
3.3. Line Detection Phase

The line detection phase is done using three steps

Canny Edge Detection
A lower threshold of 50 and upper threshold of 150 are passed to the canny edge detection.

Set ROI
A ROI is set, which is of the shape of the trapezoid.
With the vertices as [(0,xMax), (450, 340), (520, 340), (yMax, xMax)]

Hough Lines
Appropriate hough parameters are selected and hough lines are detected



3.4. Lane detection and drawing Phase

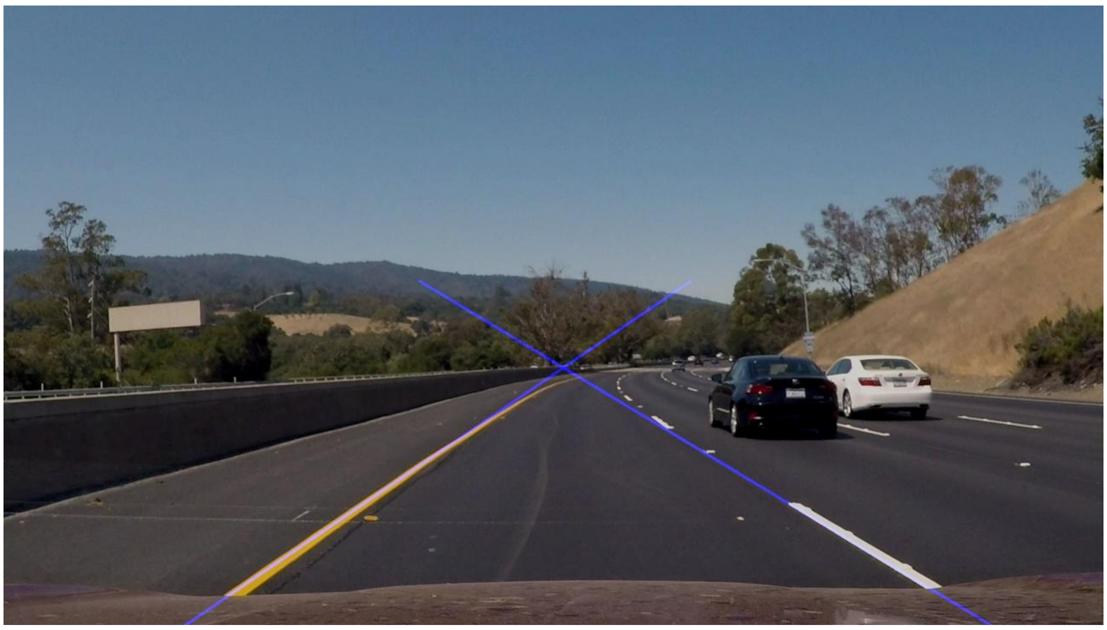
The hough lines are not continuous. Hence we need to extend them to make then into a continuous lane.

For all the given set of hough lines, the slope, intercept, and length are calculated. They are segregated to Left lanes and Right lanes, based on the slope.

The average of all the slope and intercept of the lines are calculated, given weightage to longer lines.

The right and left lines are drawn on the image

To avoid frame to frame jitter, we take the previous frame line parameters and current frame line parameters and give a weightage to the previous frame.



4. Improvements

The approach works for the current videos. In the challenge video, there are some frames where there is shadows of the trees, where the detection isn't appropriate.

Also, there are other environmental conditions where there could be tremendous improvement. Also night conditions are not handled.