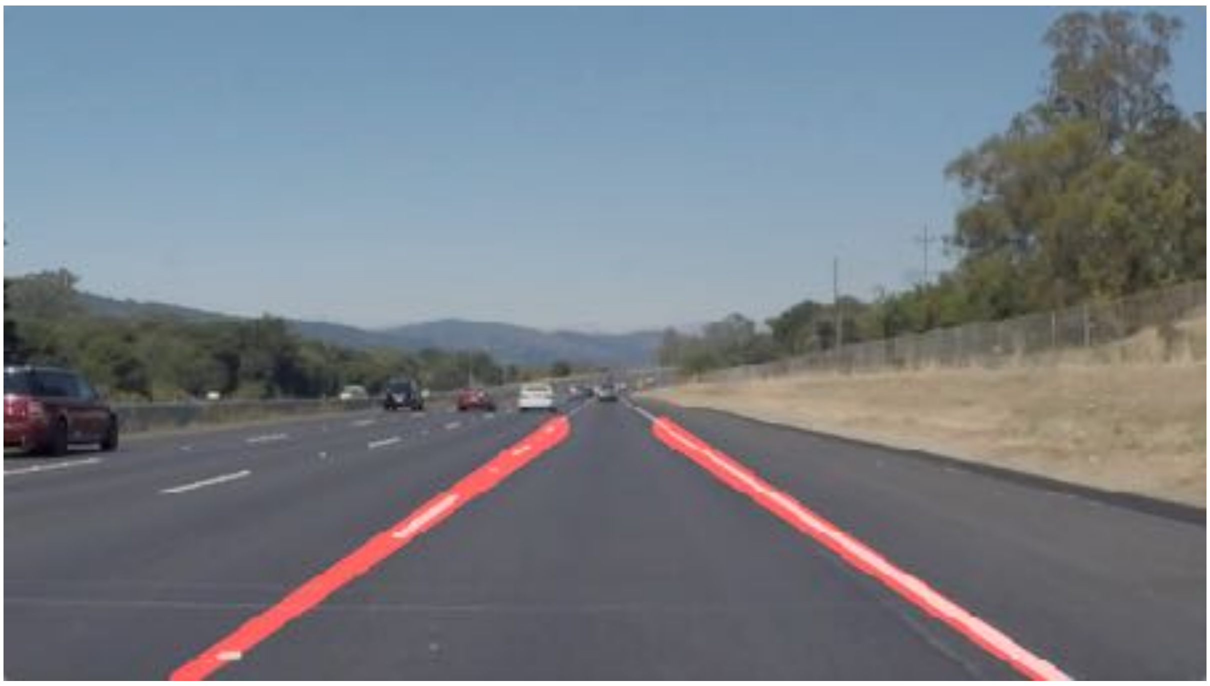
Khalil Fadhel
Udacity- Self Driving Car Course
Term 1

23 July 2017

Finding Lane Lines on the Road



This project comes as the first assignment in the self driving car course on coursera. The objective of this project is to detect lane lines on the road. In the first part we aim to detect the lane lines from an image taken from a camera from the car perspective. In the second part we apply the same algorithm on a video.

For this project we used **Python** as a programming language and **Open CV** as a library for image treatment. **Numpy** and **Matplotlib** was used as python libraries to handle image matrix and handling image files respectively. We finally used **FFMPEG** to handle video files.

We explain first the pipeline used to detect lane line on an image.

Lane Line Detection Pipeline

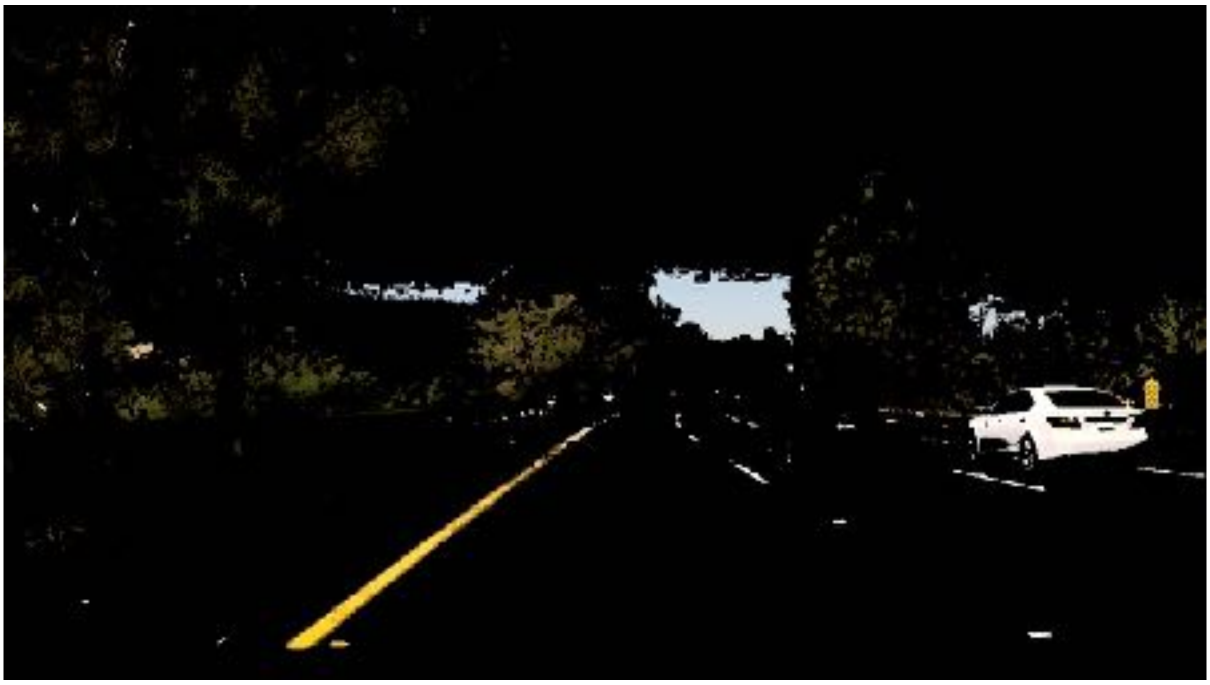
Introduction

The provided test images represent different conditions of the road. The shape of the lane lines differs, we have straight lines as well as dashed ones. The color differs as well between white and yellow lines. One other thing to notice the presence of noise that needs to be handles.

Color selection

To detect lane lines we start by extracting yellow and white colours. To do so, we transform our image from RGB colour space to HSL. The use of HSL helps selecting colours using their hue and saturation while ignoring their brightness. This comes handy in our detection of yellow lines while having shadow or a non uniform brightness on the line. To detect white lines we only use the lighting parameters.

As you can see in the image below, we already managed to remove most of the unwanted objects of the image.



Reducing Noise

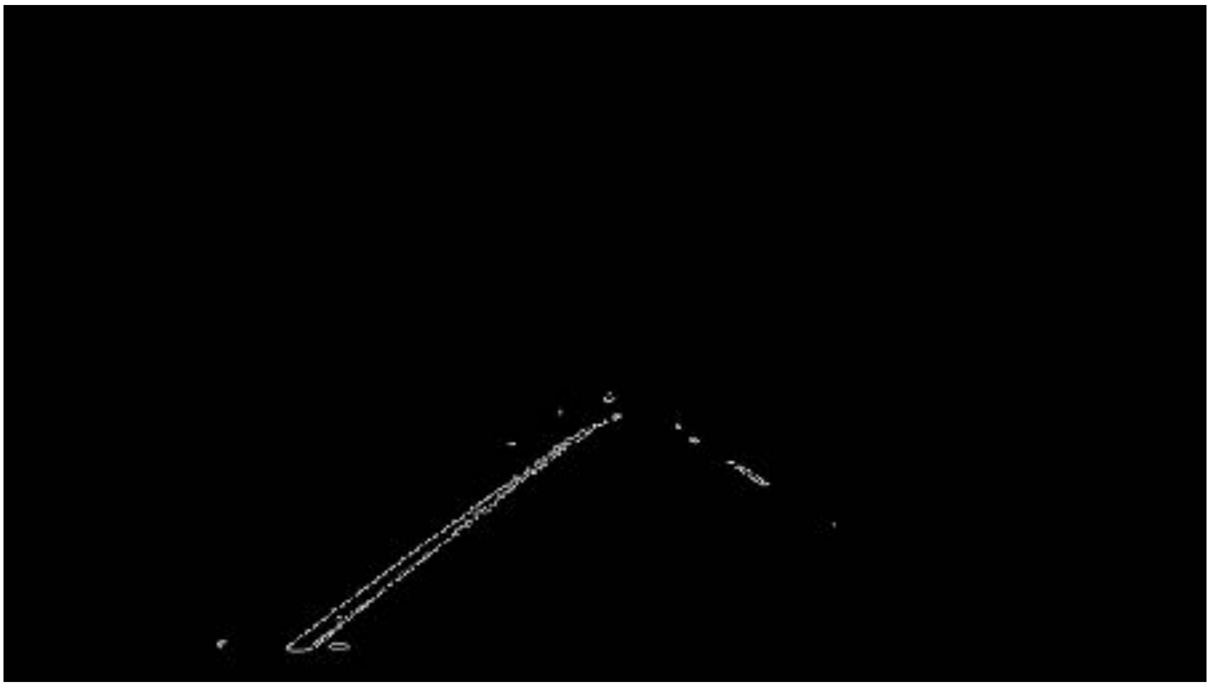
We apply a 3 by 3 gaussian blur on the image to reduce noise that may be mis-interested as lines.

Region of interest selection

The lanes appears in the same region of the screen in all the images. We use a region mask to hide the unnecessary area. As we can see in the following picture, we successfully kept only lane lines in the picture.

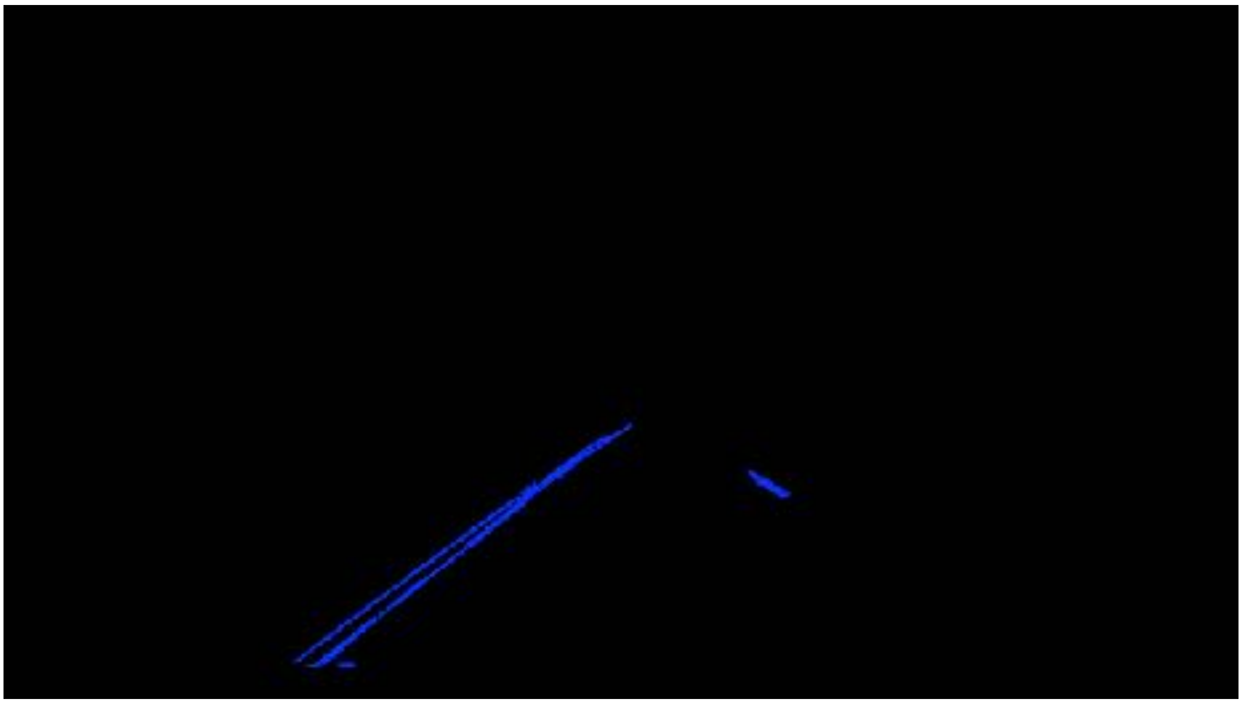
Detecting Edges

We use OpenCV's Canny algorithm implementation to detect edges. We turn first our image to gray scale. The result of the canny algorithm is a binary image representing the lines in the picture using the image gradient.



Detecting Lines

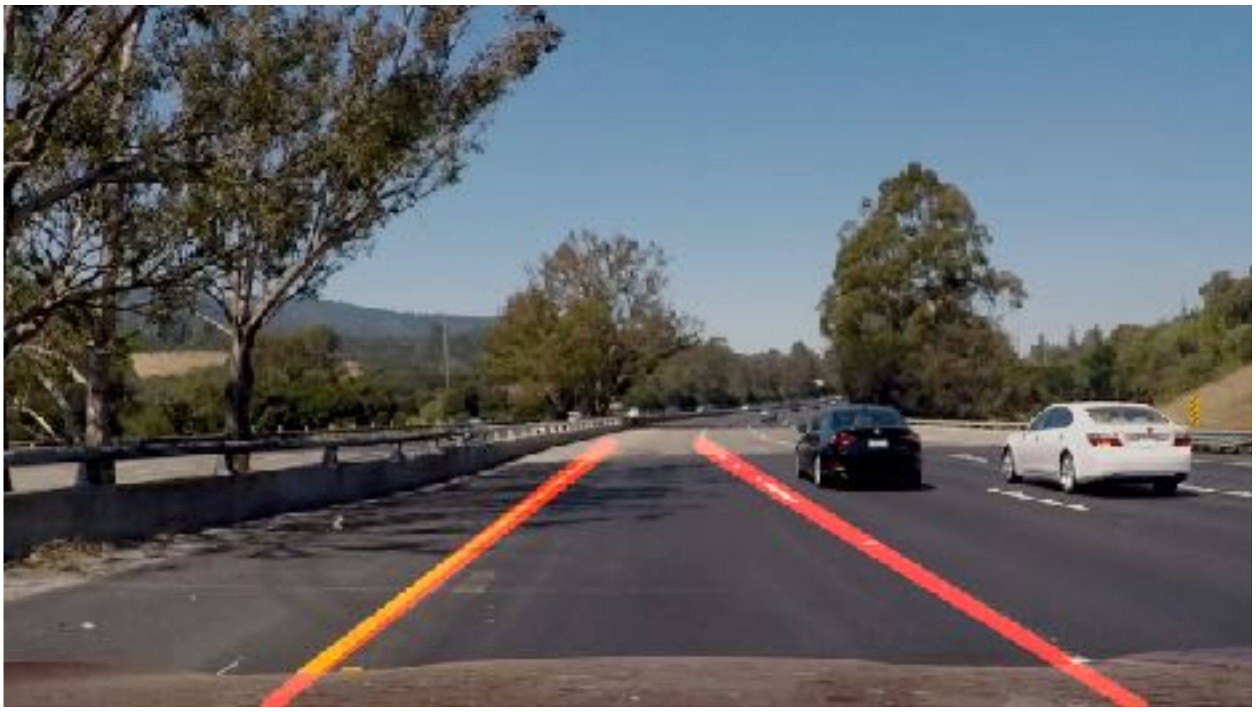
Once we successfully detected edges using Canny algorithm, we use OpenCV's implementation of Hough Transform to detect lines. The hough transform method return a list of lines.



Averaging Lines

In this part we try to average all the lines we got from the hough transform to get our left and right lane line. To do so we start by splitting our lines into two categories: right side and left side. To detect whether a line belongs to the right or left lane, we calculate his direction. For further check and removing noise, we also check his position in the image.

For the second part of averaging, we extract the points composing the lines and do a linear regression to detect the best line fitting through the points collection.



We managed to successfully detect lane lines on an image. We apply the same algorithm on the video stream and we the attended results.

Conclusion

In this project we were able to use OpenCV to detect lane lines. Using the challenge video, we were able to improve our algorithm using HSL colour transformation and to handle different video stream resolutions. We need to improve our algorithm to handle detecting lane lines in different conditions where the region selection may fail, turns as example. We can improve averaging the detected lines by hough transform by prioritise longer lines in our liner regression fitting.