Finding Lane Lines on the Road

---

1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

My pipeline consisted of multiple steps:

INPUT: The pipeline takes **an image as in input**. This image may be an isolated image or a single frame of a video. Here are example inputs to the pipeline.

1-First step**: Convert RGB image to grayscale** and apply a Gaussian blur to the grayscale images as shown:

2-Then **apply Canny transformation** to find edges on the blurred image using the parameters:

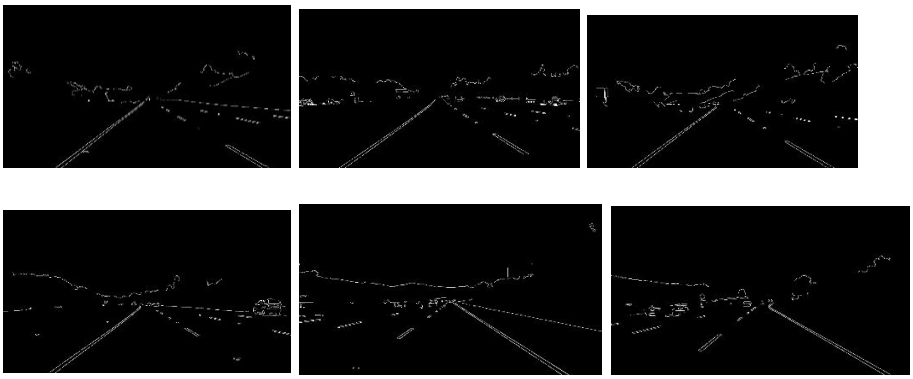
low threshold: 80

high threshold: 240

Algorithm neglects all edges below 80 and only detects:

-strong edges above high threshold

-between the two numbers only if they are connected to strong edges

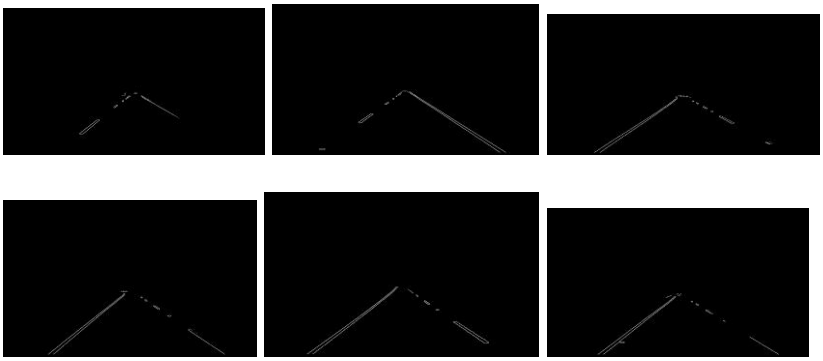


3- Then set the area of interest to only the area we are interested in.

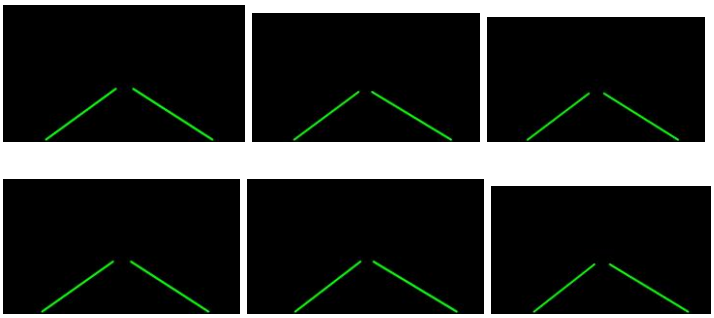
This differs from video size to another.

So I used (130,500),(450, 310), (470, 310), (900,500) for the images and the two videos .

And used (190,680),(580, 400), (640, 400), (1100,680) for the challenge video.



4- Then use hough_lines function to **find lines from the edges** in our area of interest and use these lines to **draw only 2 solid lines** that represents all lane lines.



5- Now we have an image of only lines**. Draw these lines in the original RGB image.**

Improving draw_line() function:

We want to draw 2 lines: one representing positive slope lines and the other representing negative slope lines

To draw a solid line, we should have 2 points (x1, y1), (x2, y2)

Let's use avg_x1 as the average of all X1s in the lines we want to replace with a solid line. And so avg_x2, avg_y1, avg_y2 for both positive slope line and negative slope line.

Since slope = ((y2-y1)/(x2-x1)), we have an equation of 5 variables. If we have 4, Calculate the fifth

We have average of slopes, average of x(s), average of y(s), one y point since we know exactly the y area of interest.

So we can calculate the corresponding x from the slope equation

slope = (avg_y1-y1)/(avg_x1-x1)

(avg_x1-x1) = (avg_y1-y1)/slope ,, (avg_x2-x2) = (avg_y2-y2)/slope

x1=avg_x1 - (avg_y1-y1)/slope ,, x2=avg_x2 - (avg_y2-y2)/slope

Let's set the y area of interest from 330 to 530

So we have 4 points (x1_pos, y1_pos), (x1_neg, y1_neg), (x2_pos, y2_pos), (x1_neg, y1_neg)

y1=y1_pos = y1_neg = 330 , y2=y2_pos =y2_neg = 530

So apply following equation 4 times: x=avg_x - (avg_y-y)/slope

Then draw 2 lines. Each line is drawn using 2 points (x corresponding to y1 , y1),(x corresponding to y2 , y2)

2. Identify potential shortcomings with your current pipeline

One potential shortcoming would be what would happen when the size of video changes, so each video size has a different area of interest.

Another shortcoming could be the shadows in the challenge.

Third one is drawing a curved line that represents curved lane lines in the road.

3. Suggest possible improvements to your pipeline

A possible improvement would be to make the area of interest dynamic so that we mustn’t change it every video.

Another potential improvement could be to draw more than a line or a polynomial to act as a curve.

Files (Images and videos):

test_images: This folder contains the six example images and 3 images from the challenge video.

test_images_output : This folder contains the six images after drawing solid lines.

test_video_output: This folder contains the 2 videos and the challenge video.

Each_step_output: This folder contains six subfolders each of them contain the six images after one step of the pipeline.