Finding Lane Lines on the Road

Finding Lane Lines on the Road

The goals / steps of this project are the following:

- Make a pipeline that finds lane lines on the road
    - Reflect on your work in a written report

Reflection

1. The pipeline

My pipeline consisted of 7 steps:

1. Convert image to grayscale:
This was done in order to reduce the image from a length x breadth x n_colors to just length x breadth
This effectively reduces the data by a third and allows faster computation.
Considering that most of the work required edge detection, this loss of data isn't so significant.

2. Gaussian Blur the image:
In order to remove artifacts like sudden changes in the image, I used Gaussian Blur to smooth out the image.

3. Canny Filters:
Canny filter was used to get the edges in a binary form

4. Mask the non-interesting areas:
The non-interesting areas were masked off

5. Hough Transforms:
Were used to get the lines in terms of `x1, y1, x2, y2`.

6. Calculate the average left and right lines

The utility function slope was used to calculate the slope.
The lines with slope less than -0.4 are the lines of the left lane.
The values of x1, y1, x2, y2 were added and averaged.
Values with slope greater than -0.4 and less than zero were too flat to be lane lines and hence were ignored.
Same thing was done with the right lane with the condition: slope>0.4
Note: infinite slope condition was avoided by ignoring all such cases

7. Extrapolating the averaged lines:
First the slope m was calculated for each lane
Then the equation:

```
y1_ =left_y1 + m*(x1_ - left_x1)
```

was used to calculate y1_ at different values of x1_

Different values of `x1_/x2_` allow us to extend the lines in different directions.
Basically, we need the value of `y1_/y2_` at different values of `x1_/y2_`
For the left lane, I used:

- `x1_ = 0` (and) [from the leftmost side]
- `x2_ = imwidth//2` [to the middle of the image]

For the right lane, I used:

- `x2_ = imwidth//2` [from the middle of the image]
- `x1_ = imwidth` (and) [to the rightmost side]

All the above steps were put in the process_image function and tested on the test images.
The results were:

2. Potential shortcomings with the current pipeline

1. I strongly suspect that I have implemented a very inefficient technique for lane detection. It will work for preprocessed videos, but processing it for real time videos might be too computationally inefficient.
2. Another shortcoming might be that I am using hand tweaked values for the slope cut off for the lane detection. This could fail at some edge cases.

3. Suggest possible improvements to your pipeline

1. The pipeline could be made more efficient to allow quicker real time computation.
2. The values for slopes could be modified with neural networks (perhaps) to allow more accurate values.