SDCND

Project #1 Writeup

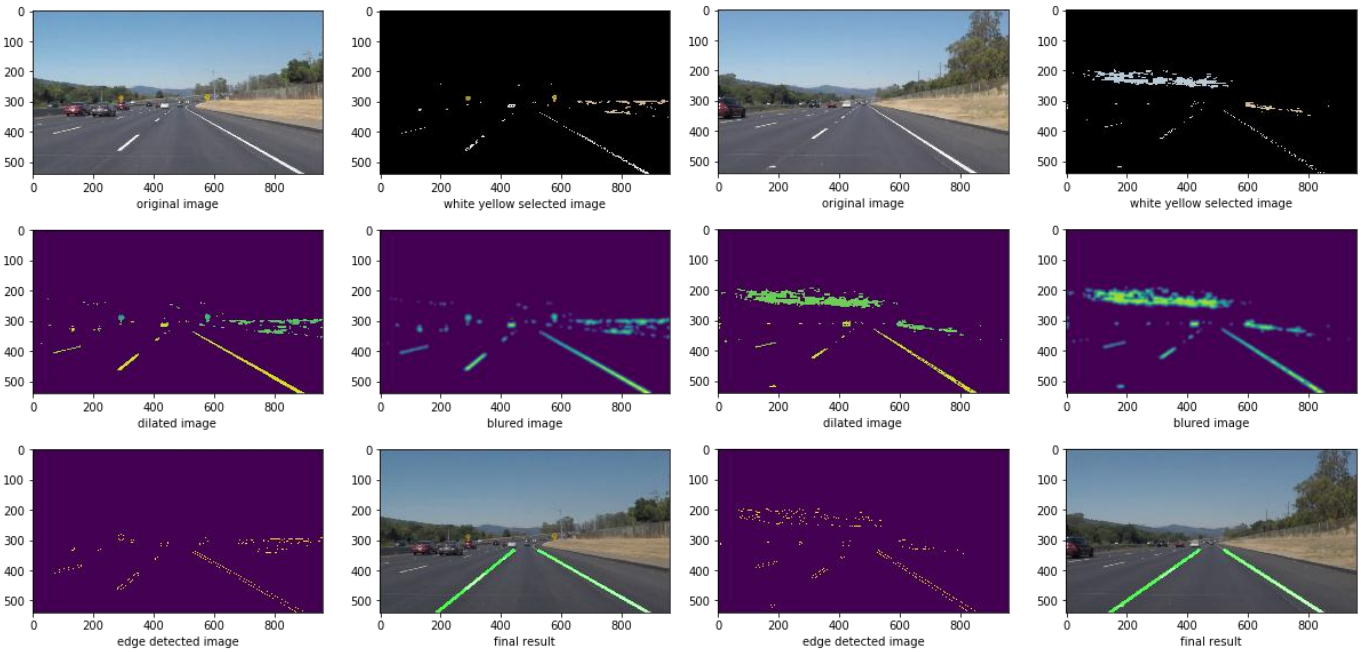
Pipeline description

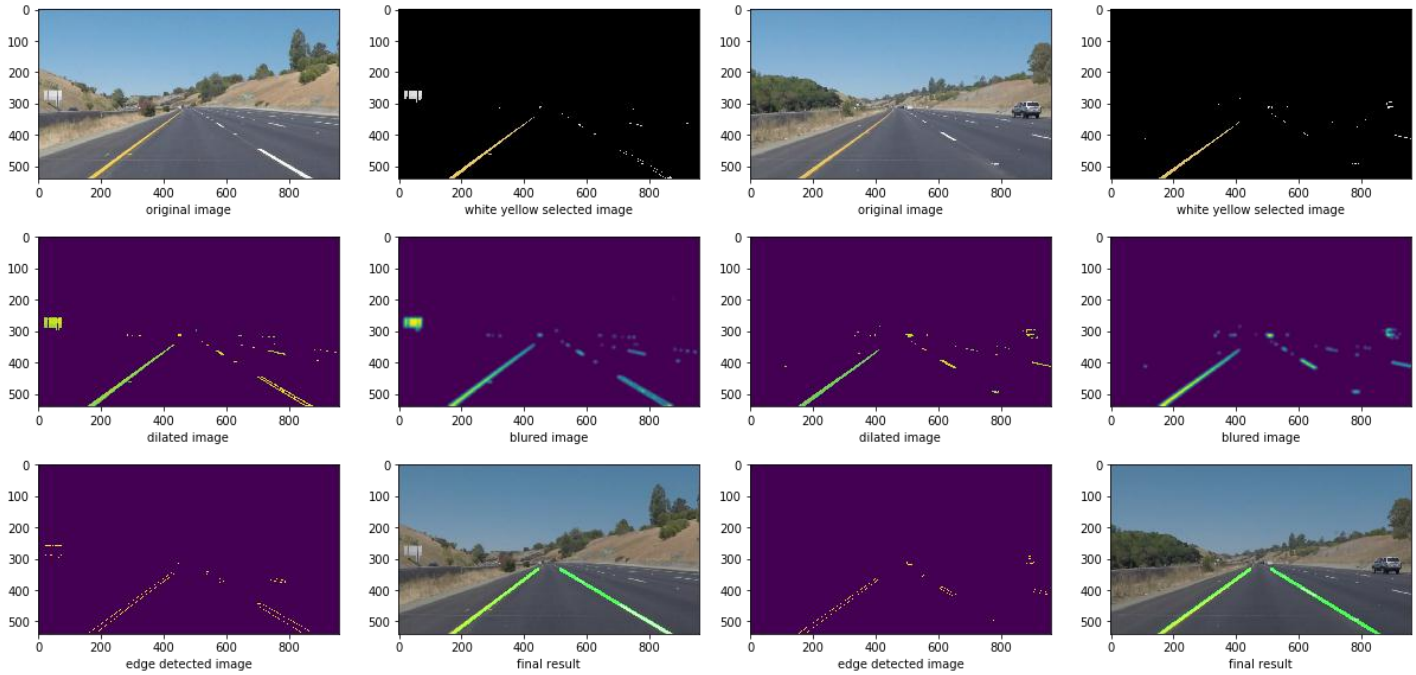
For this project I used opencv library for finding lane lines on the road.

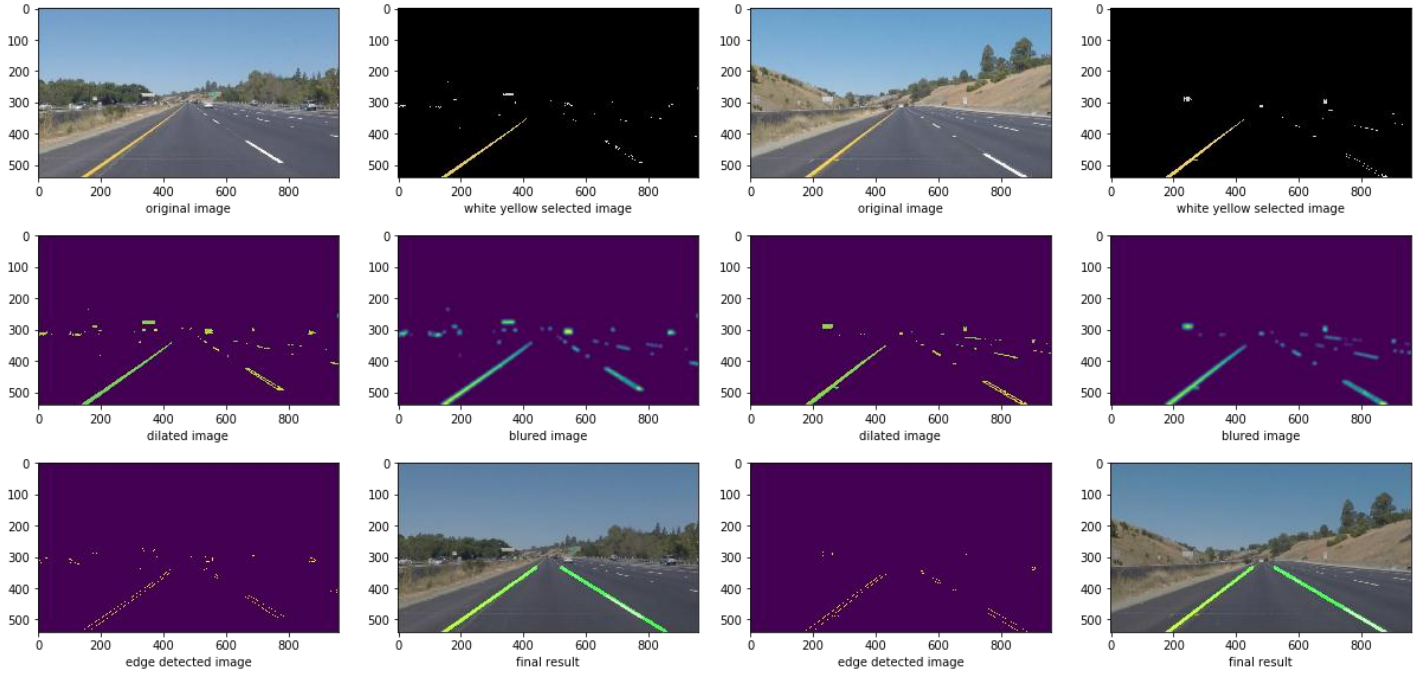
My pipeline contains following steps:

1- Color selection: for color selection I used HLS color space. I did this because performance of RGB image for challenge video was not good. But by this transformation I got good performance. After transformation I selected white and yellow colors in image using cv2.inRange() command and tuned high and low threshold for this selection.
2- Gray scale: then I converted result of previous step to gray scale image.
3- Dilate: I used cv2.dilate() function to increases the white region in the image.
4- Gaussian blur: then I used cv2.GaussianBlur(), a Gaussian noise kernel to smooth the image.
5- Canny edge detection: then used cv2.canny() to detect edges in the image.
6- Region of interest: after that I selected a region in image which the probability of lane lines in that is more than other regions.
7- Hough transformation: then I used hough transformation to find lines in the RoI. In this part I tuned parameters of hough transformation to have good performance.
8- Then I merged original image and image with drawn lane lines together.

By applying above pipeline on images, results are as follow:




original image
white yellow selected image
original image
white yellow selected image
dilated image
blured image
dilated image
blured image
edge detected image
final result
edge detected image
final result


original image
white yellow selected image
original image
white yellow selected image
dilated image
blured image
dilated image
blured image
edge detected image
final result
edge detected image
final result

In order to draw a single line on the left and right lanes, I modified the draw_lines() function by separated lines based on their slope to left and right lines. Then I calculated slope, intercept and length of each line and assigned more weights to lines with more length. I also used a 0.5 threshold for slopes to remove lines with fewer slopes and have a smoother performance and prevent drastic changes in drawn lines. After that I used mean of parameters of left and right lines and used line equation to calculate x based on y, slope and intercept:

$$y = slope * x + intercept$$

Finally, I draw each line using cv2.line() command.

After working on images, I applied my pipeline on test videos and challenge video.

Shortcoming

- Highly sensitive to color
- Highly sensitive to tuned parameters
- Sensitive to existence of objects such as cars, pedestrians, animals and so on in region of interest
- Requires hard coded regions
- Highly dependent on lane location

Possible improvements to my pipeline

- Automatic calculation of corners of RoI.
- Tuning parameters of pipeline such as hough transform parameters, edge detection, color selection, RoI and so on to improve performance and have more robust pipeline.
- Use previous frames to prevent drastic changes in slope of drawn lines.
- Use some methods to draw curves not only lines on lane lines.