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. Models. Pipeline Description.

I've tested four models.
The model_3 get the best results than model_4, but the model_3 fails in the optional challenge.

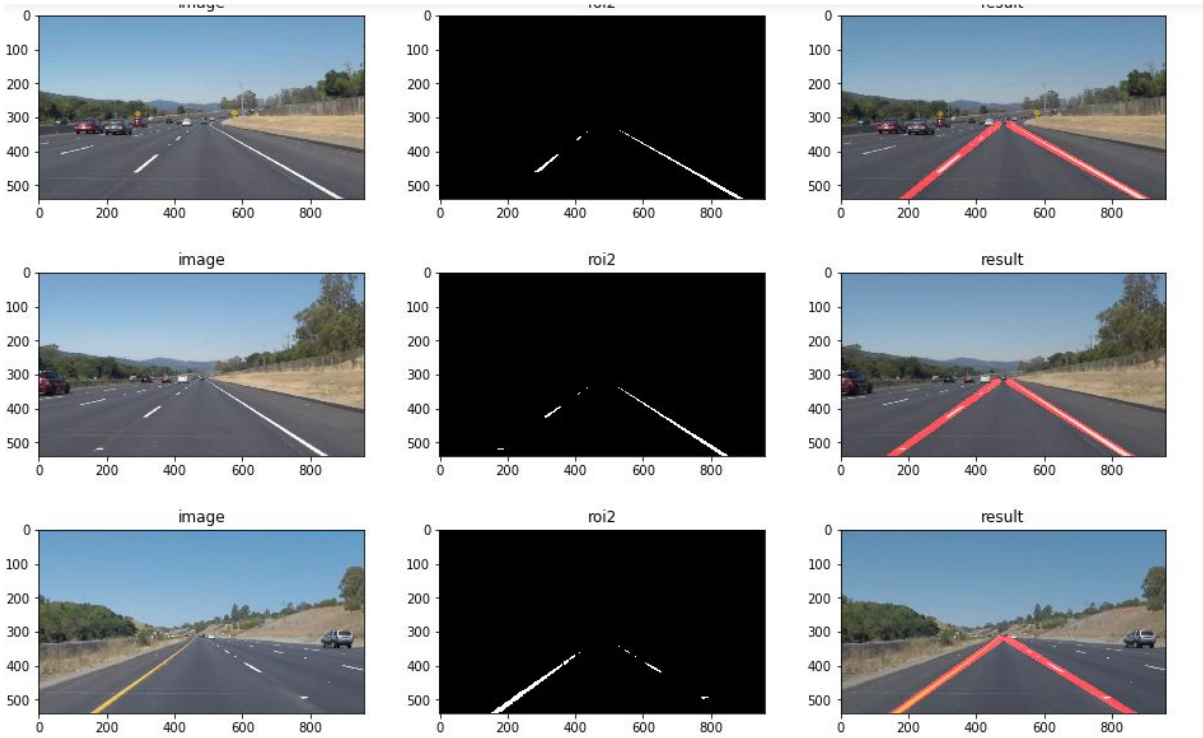
The model_4 follows the steps than we've seen in the lessons.
The model_3 consists of the following steps.

Step 1: Object Tracking
We need to extract two features of the image: white lines and yellow lines.

I've converted the images from RGB space to HSV space.
I've created two masks for white and yellow lines.
I've applied bitwise_or to the masks.

In Wikipedia article about HSV, we can read: "HSL and HSV models are based more upon how colors are organized and conceptualized in ***human vision*** in terms of other color-making attributes ". In the same way, if we visualize the hidden layers from a CNN network, we can observe similarities with ***human vision.***

Refs:
http://docs.opencv.org/3.2.0/df/d9d/tutorial_py_colorspaces.html
https://en.wikipedia.org/wiki/HSL_and_HSV
https://en.wikipedia.org/wiki/Convolutional_neural_network



Step 2: ROI
I define and apply region of interest.

Step 3: Find Lines segments
I use probabilistic Hough transform to find lines segments.

In order to draw a single line on the left and right lanes, I test several algorithms from @ypwhs and @eosrei with different parameters. I get better results using @ypwhs algorithm. However, this algorithm fails in the last challenge.

Refs:
1: https://github.com/ypwhs/CarND-LaneLines-P1
2: https://github.com/eosrei/CarND-P01-Lane-Lines

Video Results:

The left and right lane lines are annotated throughout almost all of the video.
Annotations are solid lines

Image: Annotated frame from Video “solidWhiteRight.mp4”:

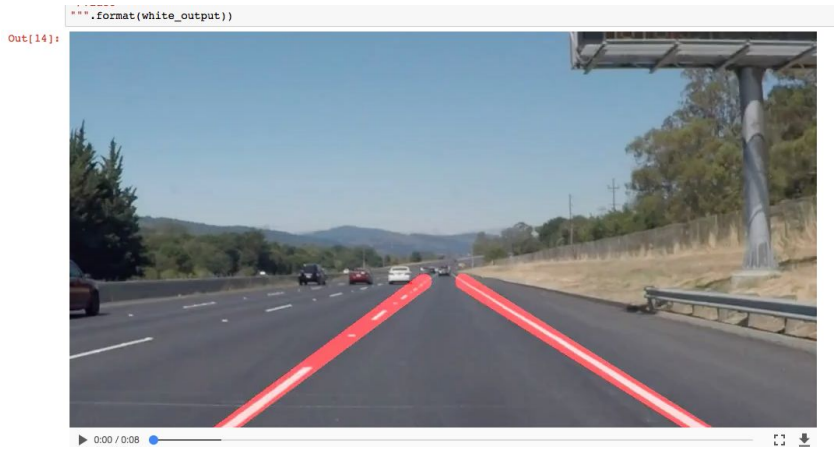
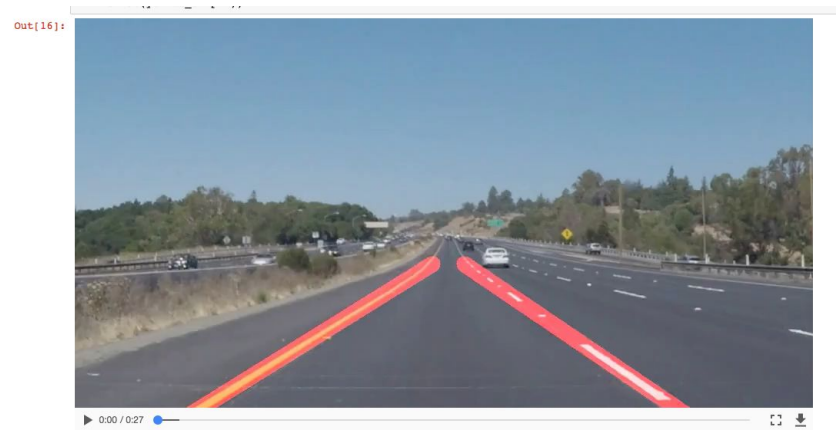


Image: Annotated frame from Video “solidYellowLeft.mp4”:



Model code:

```
def process_image_3(image, file):
    global height, width
    height = image.shape[0]
    width = image.shape[1]
    interest = np.array([[0, height], [width*3/8, height*5/8], [width*5/8, height*5/8], [width, height]], np.int32)

    hsv = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)
    yellow = cv2.inRange(hsv, (20, 80, 80), (25, 255, 255))
    white = cv2.inRange(hsv, (0, 0, 180), (255, 25, 255))
    gray = cv2.bitwise_or(yellow, white)
    roi2 = region_of_interest(gray, [interest])
    lines = hough_lines_ypwhs(roi2, rho, theta, threshold, min_line_length, max_line_gap)
    result = weighted_img(image, lines, 0.9, 0.9)
    return result
```

---

2. Identify potential shortcomings with your current pipeline

Sharp changes of light, night drive, reflections, rain, ...
In this video from **drive.ai,** we can see the real challenges than the current model won’t get pass.

https://www.youtube.com/watch?v=GMvgtPN2IBU

We are extracting features with manual engineer. This methodology is not adaptive to ***environment changes.*** If we use manual engineer to extract features, we will observe similarity shortcomings to ***environment changes*** in: speech recognition or machine translation.

3. Suggest possible improvements to your pipeline

A great improvement would be to use Deep Learning. Using DL with get automatic extraction of features.

http://selfdrivingcars.mit.edu/