Pipeline description and test on images

Basically, pipeline consists of the following steps:

1. Reading image
2. Making a copy of the image and convert it to grayscale
3. Finding edges on the gray image using Canny filter
4. Selecting ROI on Canny image
5. Finding lines on ROI masked Canny image using Hough transform
6. Putting lines found on the initial image

There are several shortcomings of this approach:

1. We have to tune parameters of Canny and Hough transform for every set of images to discard false lines. In some cases these parameters may be not suitable for images with different quality, contrast, etc., so the approach is not quite robust to variations in image properties
2. To select ROI we have to know position of the camera with respect to lanes and it has to be rather narrow region otherwise we could track the wrong lines:

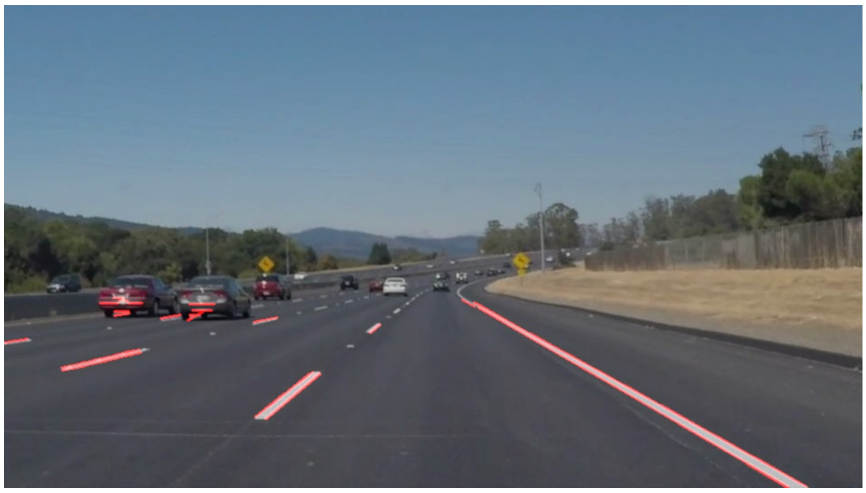
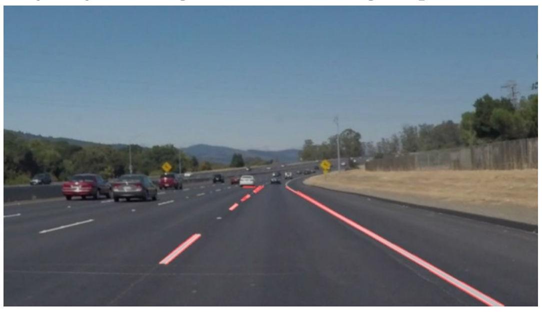
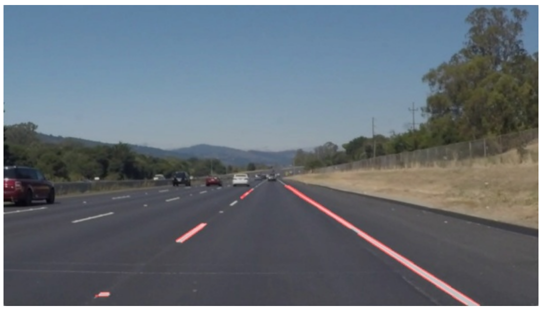
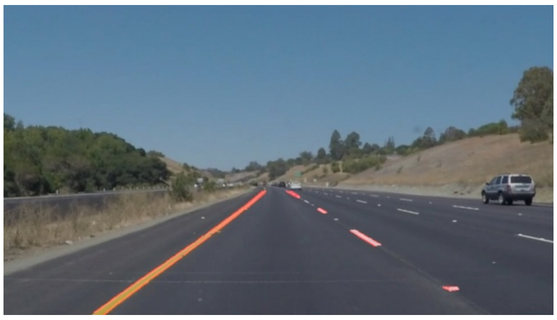


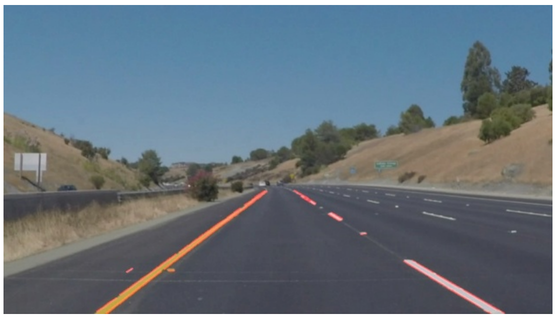
Fig.1. Too wide ROI example.

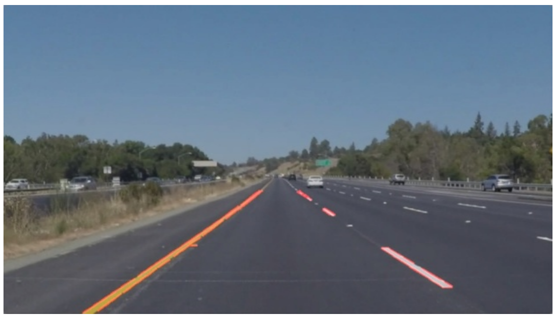
Anyway, for a given set of images quite satisfactory result was achieved:

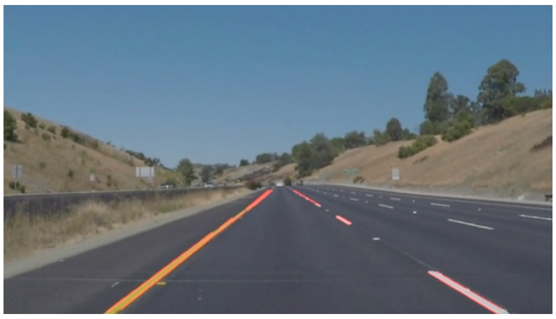












The pipeline was “packed” in **process_image** function for further use in video processing.

Improvement for line interpolation.
To get solid lines on video the following pipeline was used:

1. Reading image
2. Making a copy of the image and convert it to grayscale
3. Finding edges on the gray image using Canny filter
4. Selecting ROI on Canny image
5. Finding lines on ROI masked Canny image using Hough transform
6. Defining lines with positive and negative slopes and dividing the set of lines into two groups by this feature.
7. Filtering outliers in each group by slope. Function is_outlier() was used for that purpose (taken from https://stackoverflow.com/questions/22354094/pythonic-way-of-detecting-outliers-in-one-dimensional-observation-data).

8. Linear regression for each group of lines (positive-sloped and negative-sloped) for finding average line for each group. From this step we get average slope and intercept for each group of lies.
9. Drawing averaged lines on initial image.

The results presented in video files in **\test_videos_output** directory.

As can be seen, for **‘solidYellowLeft’** video result is satisfactory yet ‘tremor’ in averaged line can be observed. This can be partially mitigated with further selecting lines to average on. For example, length of the line can be used as weight for its endpoints during linear regression (i.e., the longer line, the more chances for it to be the right representation of a lane).

For **‘challenge’** video the result is unsatisfactory as lanes are curved. Some different approach should be implemented.

To wrap up, the main shortcoming of the approaches listed above is that they are not robust to noise and variations in camera position. This can be eliminated if some heuristic is implemented to define for a line the probability to be a part of lane.