Project #2 Part one **Zejiang Zeng**

Lane Detection **04/05/2017**

Overview:

In this project, a simple Lane Detection is made in MATLAB to mimic Lane Departure Warning System in Self Driving Cars. The input is video from a driving car and frames are extracted from the videos. After done image processing, it will return a video with lane detection. The mainly method of the lane detection is using hough transform to determine the most consistent lines.

Lane Detection:

The basic steps of the lane detection will be introduced in details in below:

1. Image preprocessing.

Pre-processing is necessary in most cases in computer vision. And the aim of it is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for future processing.

The method used in this part is remove the noise by apply the median filter. It performs median filtering of the image. Each output pixel contains the median value in a 3 by 3 neighborhood around the corresponding pixel in the input image. In MATLAB, the syntax is B=medfilt2(A), which A is the input image and B is output image.

2. Edge detection
   There are several methods to do the edge detection, like Canny, Sobel, Prewiit etc. Edge detection is to find the boundaries of objects within images. It works by detecting discontinuities in brightness. Since the lane is white, namely it is more bright that the road, so the boundaries should be extracted by using edge detection. The methods I tested are both Canny and just vertical edges (Sobel). In MATLAB, edge detection has build-in functions which is edge function.
   To use Canny edge detection, the function is BW=edge(I,'Canny',threshold). Sensitivity threshold plays an important role here, it may determine how much the brightness change is considered as boundaries. I set the high threshold to 0.5.
   The edge detected by Canny is showed in image as below:

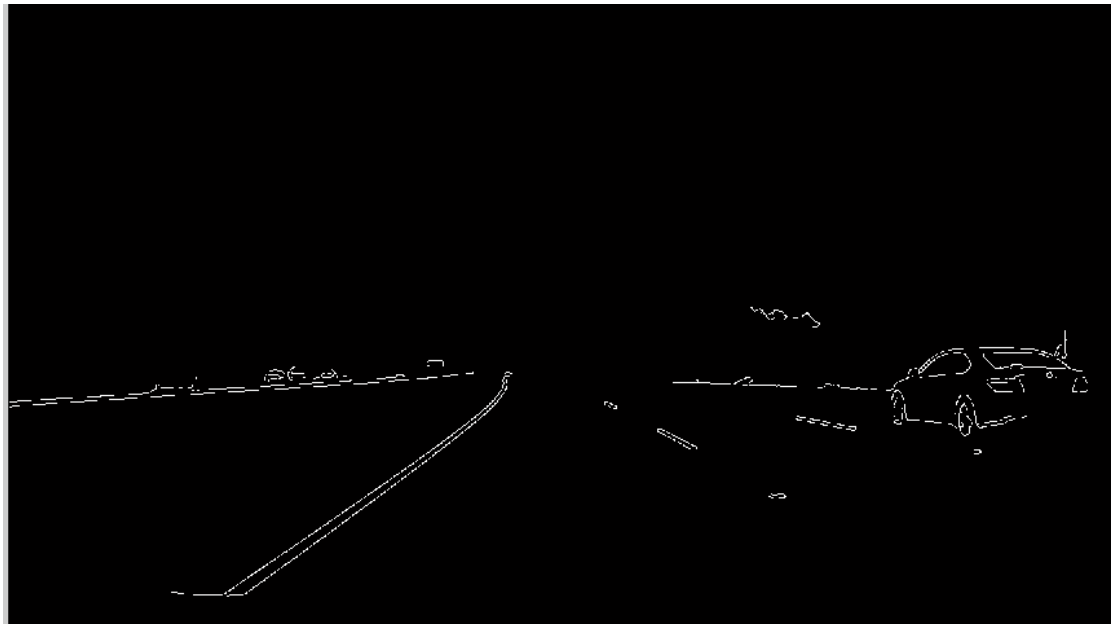

Figure 1. Canny Edge detection

Another way is to use vertical Sobel operator to just detect the vertical lines and then binaries the edge detection output. See fspecical and Imfiler for more details in MATLAB. The result for the same frame looks like figure below:

Figure 2. Sobel Vertical line detection

The Canny method gives a clear boundaries of the lane, and the vertical Sobel operation are also able to represent a clear line. In this project, Canny edge detection is used.

3. Extract the Region of Interest

As we see, not all the region of image are useful, there are trees and other cars on other lanes. We need get rid of them in order to focus more on the lane. So in my code, after tuning a little bit based on the result of Hough line detection which will introduced in next section, I mask the 1/6 to the right and 1/6 to the left out of

region of interest. And also mask out top 2/3 region of the image. As result, we only interested in the lower middle region. After we extract the region of interest, the useful part is showed as figure below:

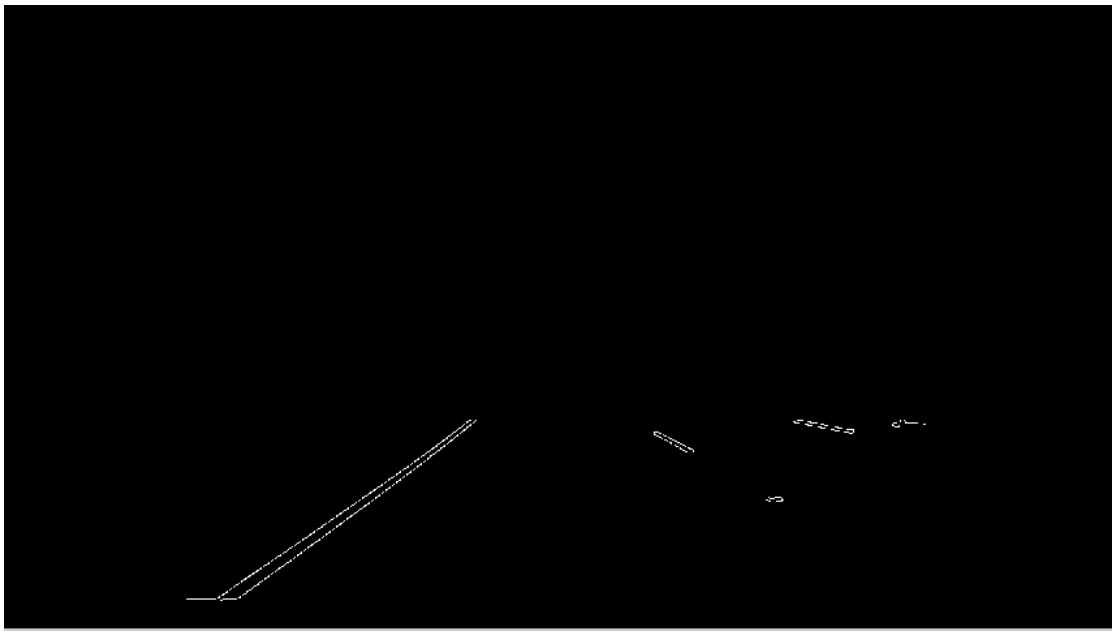

Figure 3. Region of Interest

4. **Get Hough Lines and Group them**

   The next step is to extracts line segments in the image associated with particular bins in a Hough transform. A build-in function in MATLAB will extract the start, end points, and theta, rho of the lines. The result of the hough lines detection is showed as figure below.

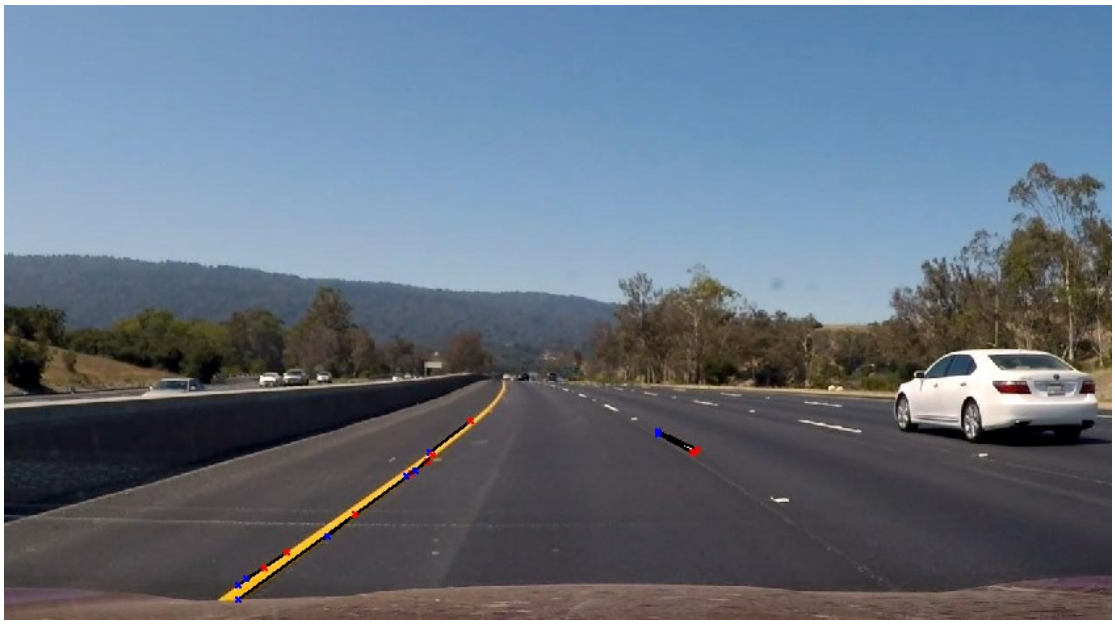

Figure 4. Hough Lines

As we can see, the lines are drawn in black. For the each road lines (right and left), there are several lines are been detected. They can be easily distinguished

intuition. If we check the theta value of each line, we can classify them into left line group if theta is positive and its value is not larger than 70. In contrast, lines in right line has a negative theta value.
Since all the detected line points are located on the road line. Thus, for the left line, I choose the highest point and lowest point and construct a straight line to approximate the road line. For the right road line, I choose the line that has the meddle theta value if there are several lines are found, and using is theta and rho with the highest, lowest points from the left to reconstruct a new line to represent the right road line. It looks like in figure 5

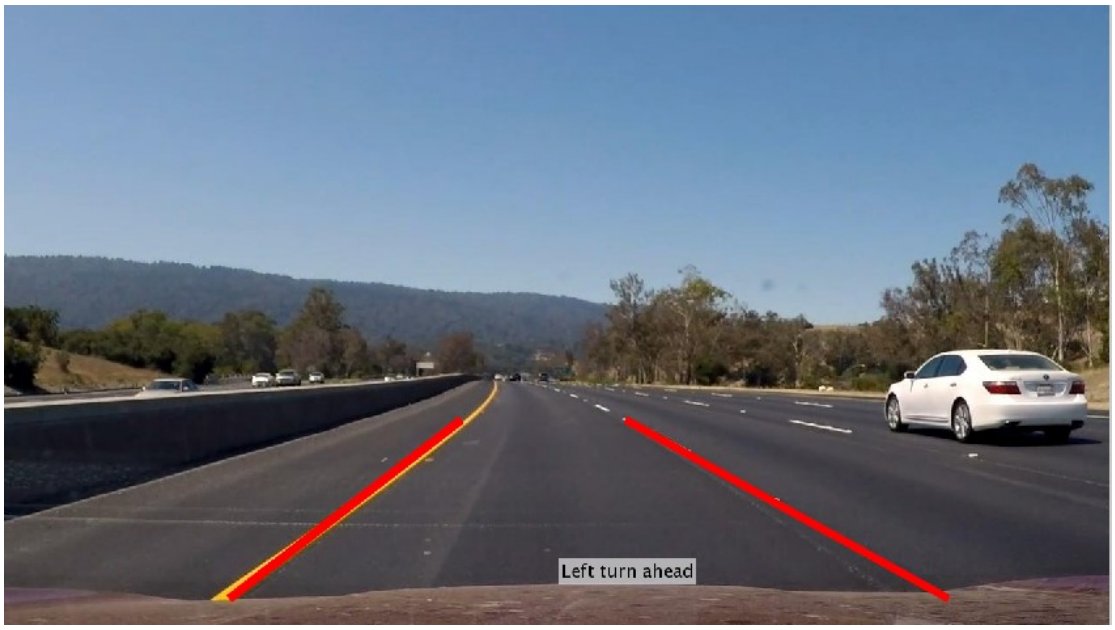

Figure 5. Approximate Road Lines

5. Predict Turns

The way I predict the turns is using central line to check if the gradient is positive or negative. Basically, it will calculate the center of the between the right and left line, if it has a negative gradient, it means it will be a turn to the left. On the contrary, it will be a right turn if it has a positive gradient. A text with the prediction of turn is added to the frame as showed in figure 5. The problem here is that this way is not robust for the every video frame. It works well for left turns and not well for right turns, since the way I approximate the right road line may has an error when trying to locate the highest and lowest points. This error may affects the gradient.

Conclusion:

Basically, my code can extract the lines accurately and the video is quite smooth when put all the frames together. However, turn prediction may not work well for right turns. It can be improved in future work.