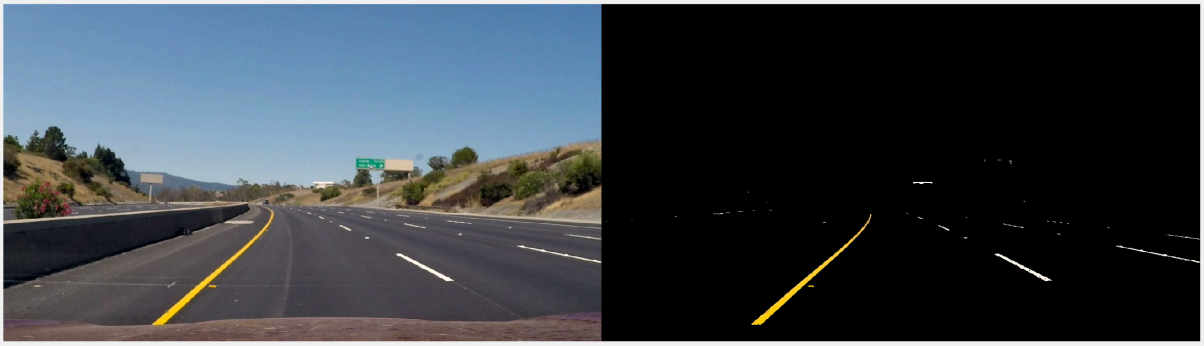
ENPM 673- Perception for Autonomous Robots
Project 1: Lane Detection
Report

Objective:
To do simple Lane Detection to mimic Lane Departure Warning systems used in Self Driving Cars.

Files attached
laneDetection.m – For project_video.mp4
laneDetectionChalleng.m – For challenge_video.mp4

Approach

Step 1: Extracting the yellow-colored and white-colored regions of the colored image.



The frames from the video are converted to HSV colorspace where it can be filtered in the hue, saturation and value map to obtain just the yellow colored and white colored as shown in the above image by defining threshold for each channel in HSV colorspace.

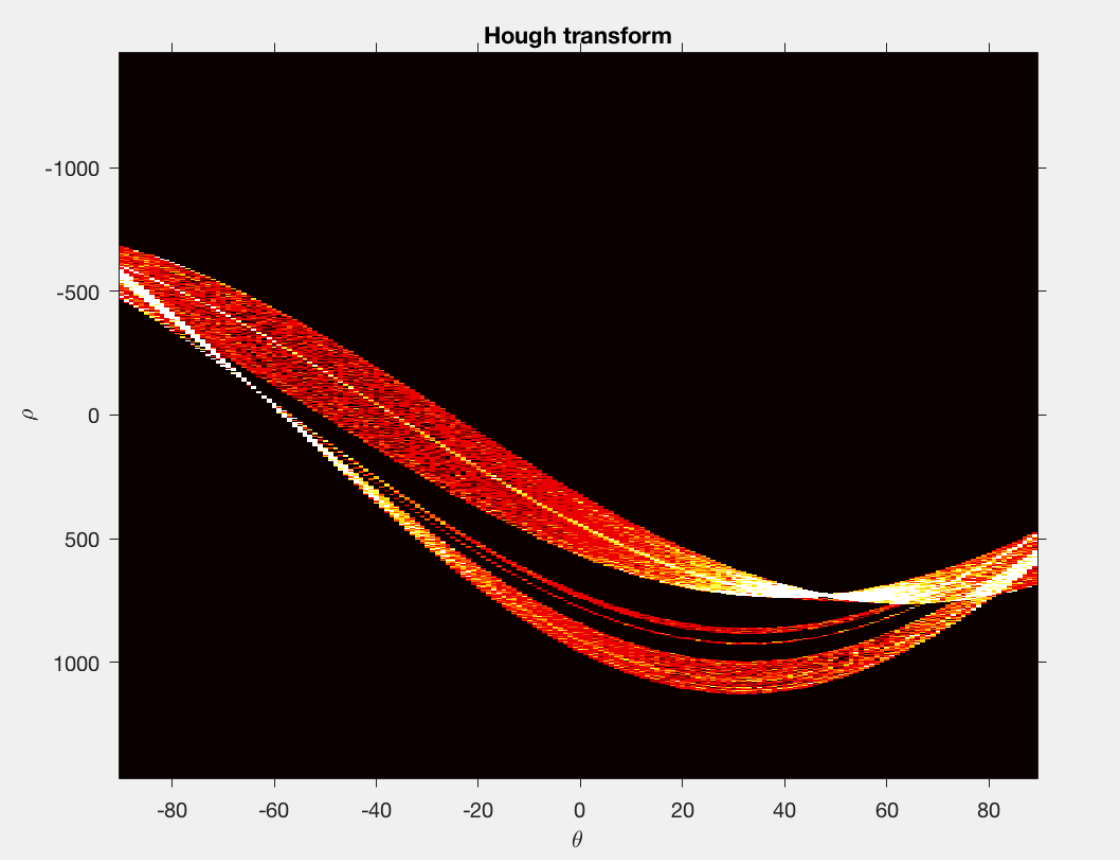
Step 2: Applying edge detection to identify the vertical edges to the previously obtained image.

The ‘sobel’ function is used to find the edges in the color filtered image from the previous step.

Step 3: Extracting the region of interest.

The ‘roipoly’ function is used to define the region of interest which gives a mask that can be applied to the image obtained from the previous step to extract the region of interest.

Step 4: Plotting hough transform for the extracted region of interest.

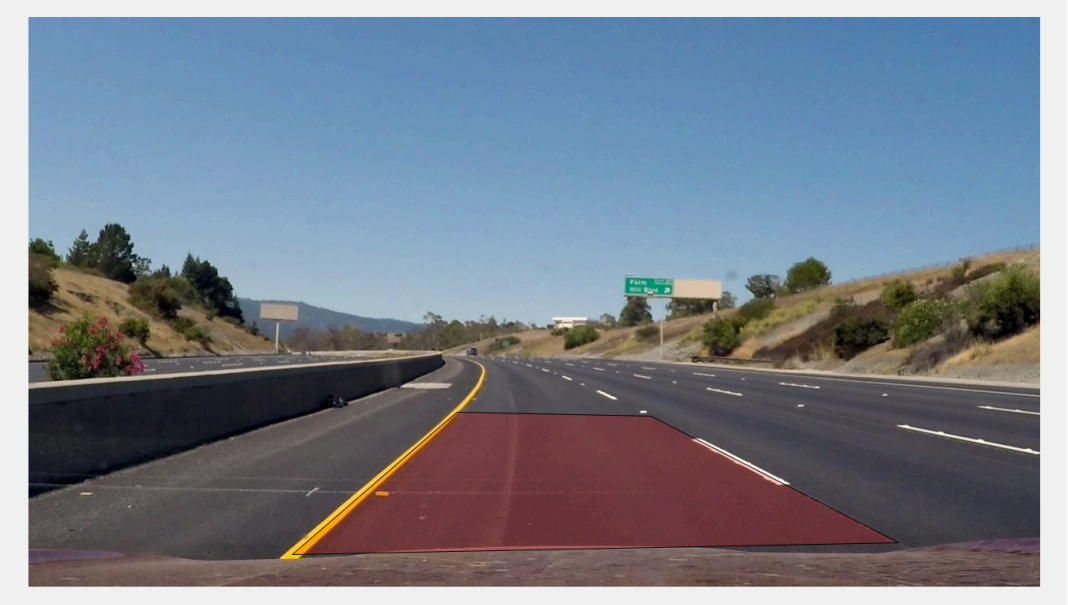


The ‘hough’ function is used to find the hough transform for the extracted region of interest which looks like the figure shown above.

Step 5: Finding the peak hough lines that maps to the lanes on either side.

The ‘houghpeaks’ and ‘houghlines’ functions are used to find the peaks in the hough transform of the image and hence obtaining lines that gives the edges in the image.

Step 6: Grouping the hough lines to obtain one line on either side of the image and extrapolating them.



The gradients of the line are grouped based on whether it is positive or negative.Thus we obtain just one line for either lanes. With the new slope for the two lines, extrapolation is done so that the visualization looks better as shown above.

Step 7: Predicting the direction of the lane.

The direction of the lane is predicted by finding the x coordinate of the intersection of the two lines. If the x-coordinate coincides with the center of the image with some marginal error window on either side the direction of the lane is 'Straight'. If the x-coordinate lies to the left of the center window the lane is turning towards 'Left'. Otherwise, the lane is turning towards 'Right'.

Challenges faced

1. The road in the bridge region has a lighter shade thus making it difficult for detecting the yellow and white markings in the lane as edges.

   Solution:
   Instead of directly applying edge detection to original image, edge detection is applied to the segmented image (the image shown in step 1 with just yellow and white-colored objects in it) so that it works irrespective of the color of the road. This is done using the following function.

```
function maskedRGBImage = createMask(RGB)
  I = rgb2hsv(RGB);
  channel1Min = 0.8;
  channel1Max = 0.2;
  channel2Min1 = 0.000;
  channel2Max1 = 0.090;
  channel2Min2 = 0.400;
  channel2Max2 = 1.000;
  channel3Min = 0.8;
  channel3Max = 1.000;
  BW = ( (I(:,:,1) >= channel1Min) | (I(:,:,1) <= channel1Max) ) & ...
       ((I(:,:,2) >= channel2Min1 ) & (I(:,:,2) <= channel2Max1)|...
       (I(:,:,2) >= channel2Min2 ) & (I(:,:,2) <= channel2Max2)) & ...
       (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
  maskedRGBImage = RGB;
  maskedRGBImage(repmat(~BW,[1 1 3])) = 0;
end
```

2. The turn prediction is sometimes oscillating between two directions as there are regions of overlap for 'Left' and 'Straight directions and 'Right' and 'Straight' directions.

Solution:
Instead of just using the direction obtained from the following function

```
function [direction,x] = predictTurn1(m,c)
  x = (c(2)-c(1))/(m(1)-m(2));
  if(x<635)
    direction = 'Left';
  elseif(x>645)
    direction = 'Right';
  else
    direction = 'Straight';
  end
end
```

The following block of code is used to check if the previous direction should be updated.

```
if(~strcmp(direction,prevDir) && ...
((center>633 && center<640) || ...
 (center>643 &&center<650)))
 direction = prevDir;
elseif(~strcmp(direction,prevDir) && ...
 ((strcmp(prevDir,'Right') && ...
  strcmp(direction,'Left')) || ...
 (strcmp(prevDir,'Left') && ...
  strcmp(direction,'Right'))))
 direction = prevDir;
end
```

3. The region under the bridge in challenge_video dataset is plagued by a shadow which makes it impossible to detect the edges.

   Solution:
   The following block of code is used to preserve the previous line coordinates until the shadow region exists assuming there cannot be sharp change in the lane position and orientation.

```
if (~isnan(x1) && abs(x(1)-x1) < 50)
  x(1) = x1;
end
if (~isnan(x2) && abs(x(2)-x2) < 50)
  x(2) = x2;
end
if (~isnan(x3) && abs(x(3)-x3) < 50)
  x(3) = x3;
end
if (~isnan(x4) && abs(x(4)-x4) < 50)
  x(4) = x4;
end
```

Result
The lane detection module is successfully built with robust turn prediction.