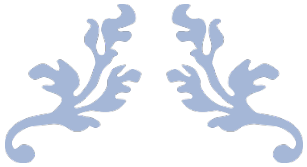
UNIVERSITY OF
MARYLAND



# PROJECT 1 - LANE DETECTION

ENPM673 Perception for Autonomous Robots

FEBRUARY 27, 2018

115617018
RajendraMayavan Rajendran Sathyam

# Contents

## Preprocessing Image

The image is first processed to remove the noise. This helps reduce the unwanted detection of edges in the future steps.

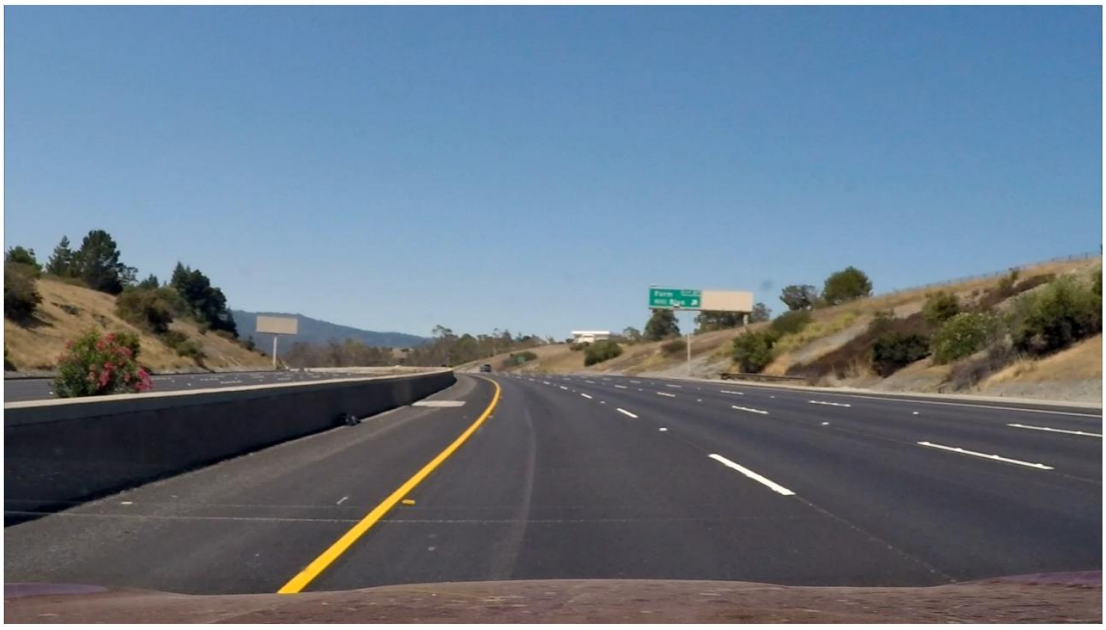


*Figure 1 Filtered frame*

```
% Filter video
motion = fspecial('motion', 2, 2);
frame = imfilter(frame, motion);
```

## Edge Detection

The image is converted to greyscale and canny edge detection is applied with a threshold of 0.5.

*Figure 2 Edges from canny edge detection*

```
% Convert to greyscale
grey=rgb2gray(frame);

% Edge detection
BW = edge(grey, 'Canny', 0.3);
```

## Region of Interest Extraction

The region of interest is the lane to be detected. Hence and mask is drawn over the binarized image of edges.

*Figure 3 Masked to get the Region of Interest*

```
% Mask an image
```

```
x=[200, 1210,725,550];
y=[675,675,450,450];
bw = poly2mask(x,y,720,1280);
BW=BW&bw;
```

# Extracting Hough Lines

## Extracting required lines from edges

Since the edge detected has two lines in the edges, it is combined by thickening the lines. After thickening the lines there is one line on either side. Then the skeleton of this thick line is extracted to get the Hough Lines center of the lane lines.

```
% thicken the lines
se= strel('disk',8);
BW=imopen(~BW,se);
BW=~BW;
```

*Figure 4 Thicken Lanes*

```
% Clean and make line thin
BW = bwmorph(BW,'clean');
BW = bwmorph(BW,'skel',Inf);
```

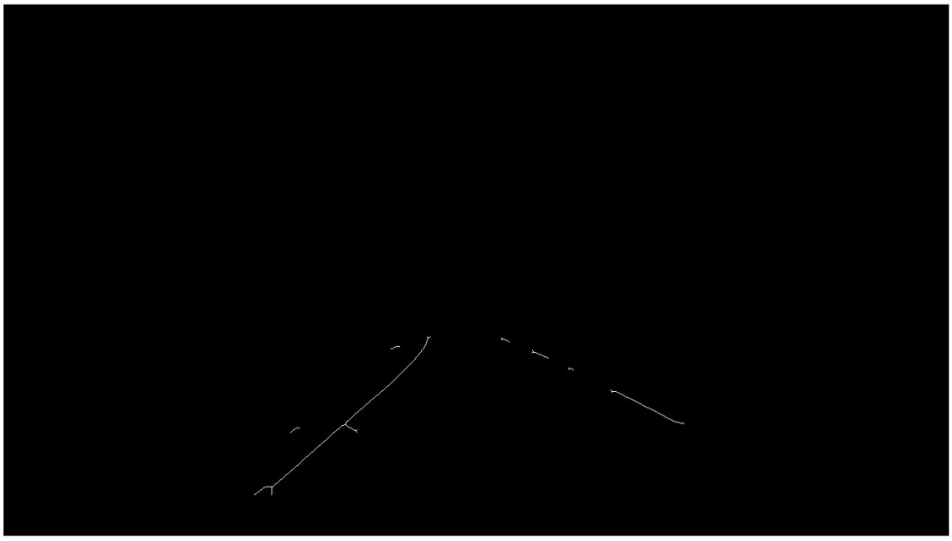


*Figure 5 Lanes skeleton*

## Applying Hough lines to the image

The extracted lines are to be drawn on the images. The lines in the left have a theta value less than 0 and the lines on the right have a theta value higher than 0.

The average slope of the left and right lane is used to predict the turning direction. If the average slope is less than -0.13, the lane turns left, if the average slope is greater than +0.13, the lane turns right, else the lane is straight.


*Figure 6 Final frame with Lane highlighted*

```
    [H,T,R]=hough(BW);

    P= houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
    lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);

    imshow(frame), hold on

    max_len=0;
    max_len2=0;
    average_slope=0;

    for k = 1:length(lines)
        xy = [lines(k).point1; lines(k).point2];

        % Determine the average slope of the line segment on left (top most
and
        % bottom most)
        if(lines(k).theta>0)
            % Determine the endpoints of the longest line segment
            len = norm(lines(k).point1 - lines(k).point2);
            if ( len > max_len)
                 max_len = len;
                 xy_longleft = xy;
            end
        end

        % Determine the average slope of the longest line segment on right
(top most and
        % bottom most)
        if(lines(k).theta<0)
           % Determine the endpoints of the longest line segment
             len = norm(lines(k).point1 - lines(k).point2);
             if ( len > max_len2)
               max_len2 = len;
               xy_longright = xy;
             end
        end
    end

    average_slope = (xy_longleft(1,2)-xy_longleft(2,2))/(xy_longleft(1,1)-
xy_longleft(2,1)) + (xy_longright(1,2)-xy_longright(2,2))/(xy_longright(1,1)-
xy_longright(2,1));
    average_slope=average_slope/2;
    text(640,360,num2str(average_slope));
    average_slope_list= [average_slope_list average_slope];

    if(numel(average_slope_list)>5)
        average_slope_list=average_slope_list(2:end);
    end

    turn_threshold = (0.13*5);
    if sum(average_slope_list)<-turn_threshold
        text(640,700,"Turning Left")
```

```
    elseif sum(average_slope_list)>turn_threshold
        text(640,700,"Turning Right")
    else
        text(640,700,"Straight")
    end

    edge_bottom=700;
    edge_top=470;

    left_p1=((edge_bottom-xy_longleft(1,2))*((xy_longleft(2,1)-
xy_longleft(1,1))/(xy_longleft(2,2)-xy_longleft(1,2))))+xy_longleft(1,1);
    left_p2=((edge_top-xy_longleft(1,2))*((xy_longleft(2,1)-
xy_longleft(1,1))/(xy_longleft(2,2)-xy_longleft(1,2))))+xy_longleft(1,1);

    right_p1=((edge_bottom-xy_longright(1,2))*((xy_longright(2,1)-
xy_longright(1,1))/(xy_longright(2,2)-xy_longright(1,2))))+xy_longright(1,1);
    right_p2=((edge_top-xy_longright(1,2))*((xy_longright(2,1)-
xy_longright(1,1))/(xy_longright(2,2)-xy_longright(1,2))))+xy_longright(1,1);


    % Plot longest left line

plot([left_p1,left_p2],[edge_bottom,edge_top],'LineWidth',2,'Color','green');
    plot(left_p1,edge_bottom,'x','LineWidth',2,'Color','yellow');
    plot(left_p2,edge_top,'x','LineWidth',2,'Color','red');

    % Plot longest right line

plot([right_p1,right_p2],[edge_bottom,edge_top],'LineWidth',2,'Color','green'
);
    plot(right_p1,edge_bottom,'x','LineWidth',2,'Color','yellow');
    plot(right_p2,edge_top,'x','LineWidth',2,'Color','red');
```