A close up of a logo

Description generated with very high confidence



Project 1 - Lane Detection

ENPM673 Perception for Autonomous Robots



February 27, 2018

115617018

RajendraMayavan Rajendran Sathyam

Contents

[Preprocessing Image 1](#_Toc507587477)

[Edge Detection 1](#_Toc507587478)

[Region of Interest Extraction 2](#_Toc507587479)

[Extracting Hough Lines 3](#_Toc507587480)

[Extracting required lines from edges 3](#_Toc507587481)

[Applying Hough lines to the image 5](#_Toc507587482)

[Figure 1 Filtered frame 2](#_Toc507587523)

[Figure 2 Edges from canny edge detection 3](#_Toc507587524)

[Figure 3 Masked to get the Region of Interest 4](#_Toc507587525)

[Figure 4 Thicken Lanes 5](#_Toc507587526)

[Figure 5 Lanes skeleton 5](#_Toc507587527)

[Figure 6 Final frame with Lane highlighted 6](#_Toc507587528)

# Preprocessing Image

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

A close up of a road

Description generated with very high confidence

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.

A screenshot of a cell phone

Description generated with high confidence

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.

A close up of a logo

Description generated with high confidence

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.

A sign on the side of a road

Description generated with very high confidence

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');