PROJECT-1: LANE DETECTION

ENPM 673 – PERCEPTION FOR AUTONOMOUS ROBOTS

PROF. CORNELIA FURMULLER

UNIVERSITY OF MARYLAND COLLEGE PARK

PROJECT REPORT BY:

ABHISHEK KATHPAL

DIRECTORY_NAME: AKATHPAL

CONTENTS:

PIPELINE USED FOR LANE DETECTION

1) Undistort the Input Image

To undistort the frame that has been extracted from video, I used the undistortimage.m file, whose link was provided in the project guidelines. This function is used to correct any distortion caused due to camera lens. To calculate the parameters which are passed into this function, I used the camera matrix and distortion coefficient. The output after using this function is shown in Fig. 1.

Fig. 1. Undistort Image

➢ Parameters passed in the function

im - Image to be corrected = frame
f - Focal length in terms of pixel units = 1.628055003e+03
ppx, ppy - Principal point location in pixels = 6.71627794e+02, 3.86046312e+02
k1, k2, k3 - Radial lens distortion parameters = -2.42e-01, -4.778e-02, -1.3138e-03
p1, p2 - Tangential lens distortion parameters = -8.7910e-05, 2.2057e-02

2) Extracting Region of Interest

Mask.m file is used to create a mask to filter out the area which cannot have any lanes. I also used gaussian filter with 3x3 kernel size and sigma equals to 1. Poly2mask which is an inbuilt function is used for this. The mask is shown in Fig. 2.

```
function masked_image = mask(image)
    x = [550, 180, 1280, 720];
    y = [450, 720, 720, 450];
    img_mask = poly2mask (x,y,720,1280);
    masked_image = uint8(img_mask).*image;
    H = fspecial('gaussian', 3, 1);
    masked_image = imfilter(masked_image,H);
end
```

Fig. 2. Mask used for further processing of image

3) Thresholding the masked image using HSV colour space

For project video, using only HSV for thresholding the lines are giving good results. The threshold function is shown below. The thresholded image is shown in Fig. 3.

```
function threshold_img = threshold(masked_image)

    hsv_frame = rgb2hsv(masked_image);
    hue = hsv_frame(:,:,1);
    sat = hsv_frame(:,:,2);
    val = hsv_frame(:,:,3);
    hsv_frame_yellow=(hue<0.2 & sat>0.4 & val>0.6);
    hsv_frame_white =(hue< 0.2 & val>0.7 & sat<0.1);
    hsv_frame = hsv_frame_yellow | hsv_frame_white;
    hsv_frame = imdilate(hsv_frame,strel('disk',2));

    threshold_img = hsv_frame;

end
```

Fig. 3. Thresholded image

For challenge video, I used Sobel filter first and then used HSV colour thresholding. The combined threshold function is shown below.

```
function threshold_img = combined_threshold(masked_image)

    gray_frame = rgb2gray(masked_image);
    edges = edge(gray_frame,'Sobel',0.03,'vertical');
    edges = imclose(edges,strel('disk',8));
    %edges = imopen(edges,strel('disk',2));
    %figure(1);imshow(opened_edges);
    masked_image = uint8(edges).*masked_image;

    hsv_frame = rgb2hsv(masked_image);

    hue = hsv_frame(:,:,1);
    val = hsv_frame(:,:,3);
    hsv_frame = hue<0.2 & val>0.7;
    hsv_frame = imdilate(hsv_frame,strel('disk',3));

    threshold_img = hsv_frame;
end
```

For challenging video, before thresholding I used top view frame. The top view frame is shown in Fig. 4. This gives better result when doing thresholding. As the lines are curved in challenge video, seeing them from top makes it easier for Sobel filter to detect vertical lines. The extra lines I used in challenge are shown below:

```
    frame = undistortimage(frame, 1.628055003e+03, 6.71627794e+02, 3.86046312e+02, -2.42

    masked_image = mask(frame);
    masked_image = topview(masked_image);

    thresholded = combined_threshold(masked_image);
    trap_Points = [180 700; 550 450; 720 450; 1280 700];
    rect_Points =[475 720;475 0;800 0;800 720];
    tform = fitgeotrans(rect_Points,trap_Points,'projective');
    thresholded = imwarp(thresholded,tform,'OutputView',imref2d(size(masked_image)));
    %figure(1);imshow(thresholded);
    %figure(1);imshow(thresholded);hold on
    lines = challenge_ht(thresholded);

```

Fig. 4. Top View Frame

```
function topview_frame = topview(image)
    trap_Points = [180 700; 550 450; 720 450; 1280 700];
    rect_Points =[475 720;475 0;800 0;800 720];

    tform = fitgeotrans(trap_Points,rect_Points,'projective');
    topview_frame = imwarp(image,tform,'OutputView',imref2d(size(image)));
end
```

4) Hough Lines

For finding the hough lines inbuilt matlab functions are used. The code is hown below:

```
function lines = hough_transform(thresholded)
    [H,T,R] = hough(thresholded);
    numberofpeaks = 100;
    P = houghpeaks(H,numberofpeaks,'threshold',ceil(0.2*max(H(:))));
    lines = houghlines(thresholded,T,R,P,'FillGap',200,'MinLength',10);
end

```

This function returns a bunch of lines, which are further divided in two groups right line and left line based on slope of lines and then from those groups, maximum length candidate is chosen. The function used for extracting maximum length line is shown below:

```
function xy_long = max_length(lines)
    max_len = 0;
    for k = 1:length(lines)
        len = norm(lines(k).point1 - lines(k).point2);
        xy = [lines(k).point1; lines(k).point2];
       if ( len > max_len)
          max_len = len;
          xy_long = xy;
       end
    end
end
```

The output obtained after getting maximum hough lines is shown in Fig. 5.

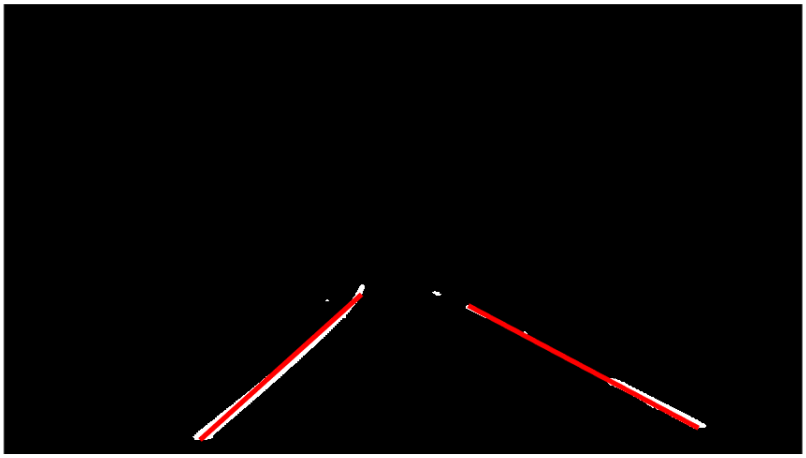

Fig. 5. Hough Lines Maximum Length

5) Predicting Turns

For turn prediction, I used the points which I get from the maximum length hough lines and then used polyfit and polyval functions to find the vanishing point. Based on the position of vanishing point with respect to centre, I predicted the direction in which car is going. The turn function is shown below:

```
function [turn_direction,ml,cl,mr,cr] = turn(xy_left,xy_right)

    p1 = polyfit(xy_left(:,1),xy_left(:,2),1);
    p2 = polyfit(xy_right(:,1),xy_right(:,2),1);
    x_intersect = fzero(@(x) polyval(p1-p2,x),3);
    %y_intersect = polyval(p1,x_intersect);
    ml = p1(1);
    cl = p1(2);
    mr = p2(1);
    cr = p2(2);
    center = 640;
    turn = x_intersect - center;
    if turn<-18
        turn_direction = 'left';
    elseif turn>18
        turn_direction = 'right';
    else
        turn_direction = 'straight';
    end
end
```

The final output is attached in Fig. 6.

Fig. 6. Final Output Image

CODE FOR LANE DETECTION

```
clc;
clear all;
close all;
tic
cd ..;
video=VideoReader('input/project_video.mp4');
n=1;

outputVideo = VideoWriter(fullfile('Output_Project_Video_1.avi'));
outputVideo.FrameRate = video.FrameRate;
open(outputVideo);


while (hasFrame(video))
    video_frame = readFrame(video);
    frame = video_frame;
    frame = undistortimage(frame, 1.628055003e+03, 6.71627794e+02, 3.86046312e+02, -2.42565104e-01,-4.77893070e-

    masked_image = mask(frame);

    thresholded = threshold(masked_image);

    lines = hough_transform(thresholded);

    lines_l = struct;
    lines_r = struct;
```

```
     slope = zeros(1,length(lines));
     i=1;j=1;
     for k = 1:length(lines)
        slope(k) = (lines(k).point2(2)-lines(k).point1(2))/(lines(k).point2(1)-lines(k).point1(1));
        if((slope(k)>0.5 && slope(k)<1) || (slope(k)<-0.5&& slope(k)>-1))
            if(slope(k) < 0)
                lines_l(i).point1 = lines(k).point1;
                lines_l(i).point2 = lines(k).point2;
                i = i+1;
            else
                lines_r(j).point1 = lines(k).point1;
                lines_r(j).point2 = lines(k).point2;
                j = j+1;
            end
        end
     end

     if(size(lines_l,2)>1)
         xy_left = max_length(lines_l);
         %plot(xy_left(:,1),xy_left(:,2),'LineWidth',4,'Color','red');
     else
         continue;
     end

     if(size(lines_r,2)>1)
         xy_right = max_length(lines_r);
```

```
        %plot(xy_right(:,1),xy_right(:,2),'LineWidth',4,'Color','red');
    else
        continue;
    end


    [direction,ml,cl,mr,cr] = turn(xy_left,xy_right);

    green_mask=poly2mask([(480-cl)/ml,(680-cl)/ml,(680-cr)/mr,(480-cr)/mr],[480,680,680,480],720,1280);
    video_frame(:,:,2)=imadd(double(video_frame(:,:,2)),double(100*green_mask));

    video_frame = insertText(video_frame,[550,600],['Going ', direction],'fontSize',24,'TextColor','black');

    writeVideo(outputVideo,video_frame);
end
close(outputVideo);
toc
```