

Finding Lane Lines on the Road

Author:Nolan57

Date:2017-3-4

Note:

All my codes(pipeline and process_image for videos) for finding lane lines on the road in:[P1-Nolan57.ipynb\(including the results of test_images\)](#)

****Finding Lane Lines on the Road****

The goals / steps of this project are the following:

* Make a pipeline that finds lane lines on the road:

```
def pipeline (initial_img,image,
              kernel_size,
              rho,theta,threshold,min_line_len,max_line_gap):
    #determine the intereting section of image
    x1=50
    y1=image.shape[0]
    x2=image.shape[1]-50
    y2=image.shape[0]
    x3=image.shape[1]/2-20
    y3=image.shape[0]/2+20
    x4=image.shape[1]/2+20
    y4=image.shape[0]/2+20
    vertices=np.array([[x1,y1],[x2,y2],[x3,y3],[x4,y4]])

    #converted the images to grayscale
    gimage=grayscale(image)
    #apply a gaussian_blur to the grayscale image
    gbimage=cv2.GaussianBlur(gimage, (kernel_size, kernel_size), 0.0)
    #define a function for determining the values of params of
    #cannny by cv2.threshold
    low_threshold,high_threshold=high_low_thresh(gbimage)
    #detect the edges by canny
    bcimage=canny(gbimage,low_threshold, high_threshold)
    #mask a intereting region on the edges to narrow the scope procssed
    #by HoughLinesP
    amimage=region_of_interest(bcimage,[vertices])
    #determine the lines on the edges inside intereting region by HoughLinesP
    himage=hough_lines(amimage,rho,theta,threshold,min_line_len,max_line_gap)
```

```
wimage=weighted_img(himage,initial_img)
```

```
return wimage
```

below is result by my pipline on the images in test_images folder:

whiteCarLaneSwitch_processed.jpeg:



solidYellowCurve_processed.jpeg:



solidWhiteCurve_processed.jpeg:



solidWhiteRight_processed.jpeg:



solidYellowLeft_processed.jpeg:



<The results of the videos will be written to “white-me.mp4” & “yellow-me.mp4” >

* Reflect on your work in a written report

Reflection

###1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

My pipeline consisted of below steps:

- 1,converted the images to grayscale
- 2,apply a gaussian_blur to the grayscale image
- 3,detect the edges by canny
- 4,mask a intereting region on the edges to narrow the scope procssed by

HoughLinesP

- 5,determine the lines on the edges inside intereting region by HoughLinesP
- 6,average the lines determined by HoughLinesP with help of **np.fitLine function**

and draw lines on original image

7, return the result image

****additional step: to manually determine proper values of params of canny and HoughLinesP, I create an interact cell by IPython.widgets on my ipynb.**

In order to draw a single line on the left and right lanes, I modified the draw_lines() function by:

1, distinguish the left lines and right lines by checking if the points of lines are less or bigger than x coordinate of middle of image width

2, exclude the left lines which's slope = 0 or < 0,

3, exclude the right lines which's slope = 0 or > 0,

4, average the left/right line by cv2.fitLine() on the points of ends of lines,

###2. Identify potential shortcomings with your current pipeline

One potential shortcoming would be:

1, no edges data to be processed when lane lines are missing at all

2, can't find the lane line on the road when other objects in the centre of interesting region

Another shortcoming could be

1, can't automatically determine the proper size, position and shape of interesting region on different images

2, can't automatically determine the proper value of params of functions on different images

###3. Suggest possible improvements to your pipeline:

add the function to automatically determine the proper interesting region by the distribution of edges

A possible improvement would be to:

add the function to automatically determine the proper value of params of canny and HoughLinesP

Another potential improvement could be to:

add the function to avoid the disturbance on finding the lane line on the road by unnecessary objects in centre of interesting region