CSE483 Computer Vision

Team members: Marwan Atef 18p8678, Mostafa Essam 18p9203

Project Phase 1 Documentation

In the first phase of the project, our goal is to detect the lanes of the road from the point of view of a mounted camera on a moving car.

Through a set of functions, we reach our desired goal. These functions are as follows:

1. region: In which we identify the region of the input image that we are interested in detecting the lanes in it.
2. average: That takes the average of a group of left lane lines and the average of a group of right lane lines, and returns the one average left lane line and the one average right lane line.
3. edge\_detection: In which we detect any edges in the input image.
4. connectPoints: In which we connect the endpoints of a lane line in order to construct a lane line.
5. display\_lines: In which we draw the two lane’s lines in magenta like color, and also fill the space between the lane lines with the color green.
6. framePipeline: Which is the backbone of this project; as it includes the stages of the pipeline from the input image to the final output. The pipeline stages are as follows

* convert the input image to grayscale.
* blur that grayscale image to help in edge detection later.
* we detect any edges in the latest image in the pipeline.
* we try to detect the lane lines (that are also edges) only from all the image edges.
* we process the hough lines and get the one average left lane line and the one average right lane line.
* Then we construct the output image or video frame by combining the drawn lanes and the filled polygon between them, and the very beginning input image.
* Finally, when the python file is done executing, two outputs are displayed:

1. The pipeline stages next to each other in order (the last image is a black blank image).

Graphical user interface

Description automatically generated with medium confidence

Figure 1 Pipeline stages

1. The desired final output with the road lane detected.

Graphical user interface

Description automatically generated

Figure 2 The desired final output

Auxiliary functions that are used in the code:

* stackImages: That takes multiple images and display them next to each other.

At the end of the project code, exists the code that handles the parameters that the bash shell sends in order to select the mode of the input whether it is an image or a video, and the path of that input.

path= sys.argv[2]  
  
if sys.argv[1] == "img":  
  
 img = cv2.imread(path)  
 img = cv2.cvtColor(img, cv2.COLOR\_RGB2BGR)  
 plt.imshow(framePipeline(img))  
 plt.show()  
  
  
elif sys.argv[1] == "vid":  
 video = cv2.VideoCapture(path)  
 frameWidth = int(video.get(3))  
 frameHeight = int(video.get(4))  
  
 while video.isOpened():  
 ret, frame = video.read()  
 if ret == True:  
 frame = framePipeline(frame)  
 cv2.imshow('Frame', frame)  
 if cv2.waitKey(1) & 0xFF == ord('s'): #press s to stop the running video  
 break  
 else:  
 break  
 video.release()  
 cv2.destroyAllWindows()

And here is the bash shell script which is used to execute the phase1.py file:

#!/bin/bash

echo "Insert 'img' for Image or 'vid' for video:"

read mode

echo "Insert the path of the file:"

read target

python phase1.py $mode $target