Homework 04: Bird's Eye Transformation and Estimate the count of the cars

Anushree Das (ad1707)

a. Abstract

The assignment has three goals:
A. Assure you can get a frame from the video.
B. Compute a Bird's eye transformation of the video.
C. Estimate the local count of the cars, using the video analytics.

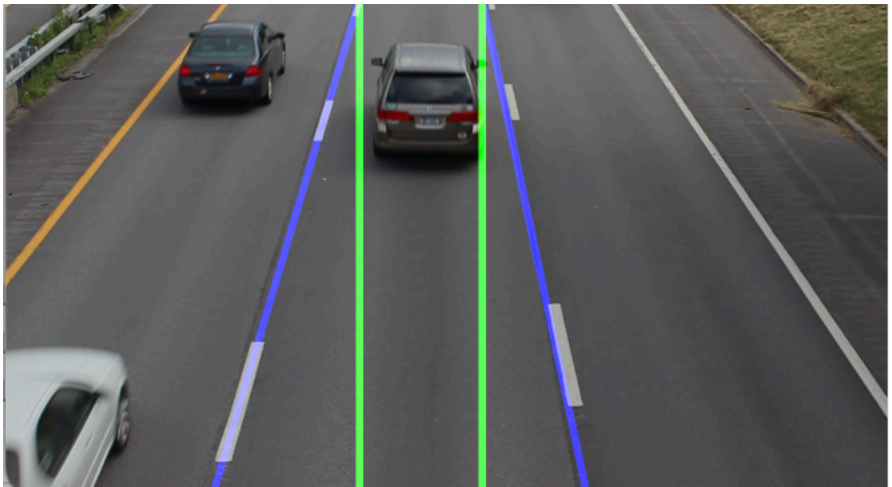
b. Overview

There are three programs: ReadVideo.py, BirdsEyeTransformation.py and MotionEstimation.py for each part, respectively. All the programs takes path to video as input after running the program. ReadVideo.py reads the first 30 frames, and save the thirtieth (30th) frame to a file in the same folder as the input video. BirdsEyeTransformation.py converts the input video into a "bird's eye" output video and displays it. MotionEstimation.py detects vehicles in the center lane of the road using the background subtraction and stores the frame numbers in which a new car was detected into a csv file in the folder where the program is.

i.Approach Used

For Part A of the homework, I used VideoCapture() from Opencv library to open the .MOV video, VideoCapture.set() to get to the 30th frame of the video and imwrite() to save the frame as image.

For Part B of the homework, I followed the tutorial on https://nikolasent.github.io/opencv/2017/05/07/Bird's-Eye-View-Transformation.html to get the bird's eye transformation. I used the lane detector code given at https://www.geeksforgeeks.org/opencv-real-time-road-lane-detection/ to get the coordinates of lines of the center lane as shown below:



The blue line shows the lane edges and the green line shows the edges it will be warped to. The getPerspectiveTransform() gives the 3x3 transformation matrix that stores the transformation values which can be applied to the whole image using warpPerspective() to get the bird's eye transformation.

As you can see in the image above, the road now looks rectangular.

For Part C of the homework, I followed the tutorial on https://www.analyticsvidhya.com/blog/2020/04/vehicle-detection-opencv-python/ to detect vehicles. I used the MOG2 background subtraction to detect foreground object mask. I applied morphology on the mask to reduce noise and cropped the mask for only the center lane of the road. A car is detected as soon as it reaches a particular line. I found the contours of the foreground object masks. The contours with area more than 3500 and that are below a particular line are considered as valid. As soon as the vehicle crosses the line, the number of contour decreases and the car is considered to be detected. The frame number is stored in a csv file.

ii.Experiment

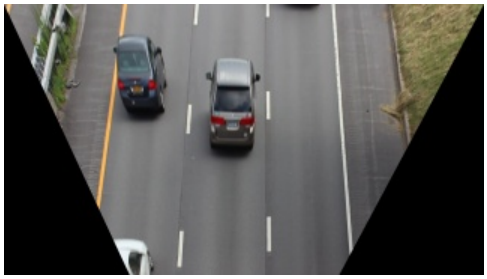
First I tried to give constant values for the source and destination points for the getPerspective(), but it did not work for all videos. Therefore, I decided to use lane detector to so that I could get points automatically.

iii.Result

The result for videos MVI_2201_CARS_ON_590_FROM_BRIDGE.MOV, MVI_2207_CARS_ON_590_FROM_BRIDGE.MOV and MVI_2208_CARS_ON_590_FROM_BRIDGE.MOV for Part A:



The result for videos MVI_2201_CARS_ON_590_FROM_BRIDGE.MOV, MVI_2207_CARS_ON_590_FROM_BRIDGE.MOV and MVI_2208_CARS_ON_590_FROM_BRIDGE.MOV for Part B:



The result for Part C:
14, 148, 249, 324, 391, 434, 523, 604, 721, 770, 936, 1001, 1221, 1351, 1434, 1516, 1777, 1809, 1886, 1954, 2012, 2062, 2140, 2271, 2349, 2389, 2484, 2549, 2631, 2757, 2844, 2961, 3011, 3174, 3215, 3300, 3338, 3426, 3532, 3657, 3758, 3843, 3900, 3983, 4143, 4306, 4483, 4551

d. Conclusions

It is possible to read .MOV video format using OpenCV. The bird's eye transformation for a video can be obtained using getPerspective() and warpPerspective() functions. The moving foreground objects can be counted using background subtraction.

e. Credits:

Part B:
Bird's eye Transformation: https://www.geeksforgeeks.org/opencv-real-time-road-lane-detection/
Lane detector: https://nikolasent.github.io/opencv/2017/05/07/Bird's-Eye-View-Transformation.html
Part C:
Car detection: https://www.analyticsvidhya.com/blog/2020/04/vehicle-detection-opencv-python/