ENPM 673, Robotics Perception
Project 3a: Car Tracking.
Due on: 11:59:59PM on Monday – May 15, 2017

Prof. Yiannis Aloimonos

1 Car Tracking - 100Pts

Tracking is an important aspect of Computer Vision and it can be helpful for *Self Driving Car* to take necessary actions by tracking other vehicles/pedestrians. In this project we aim to introduce you to various such techniques by implementing a Car Tracker. You will use **Computer Vision Toolbox** from MATLAB.

This is the first part of Project 3 and you can expect the second part in two weeks. Please note that the first part SHOULD NOT take more than 2 weeks and the next part will be designed accordingly. So PLEASE DO NOT WAIT till the actual deadline.

2 Details

Car Tracking requires *Detection* as a prior step followed by *Tracking* a Region of Interest.

Note: You are given videos from a driving car, vehicle/non-vehicle images for classification and the output should be a video submission. You need to submit your output for `simple.avi` video - DATASET.

2.1 Detection - 50 Pts

In this section you will use *Haar Cascade* from either MATLAB's Computer Vision toolbox. As a part of *Cascade Classifier* you need to generate an XML file using *positive samples* (car images) and *negative samples* (everything else). You will further use this XML file to detect cars in every frame.

- Checkout MATLAB example here
- Checkout OpenCV tutorial here

As always you can filter false positives by analyzing the bounding box properties like centroid location, aspect ratio, area etc.

2.2 Tracking - 50 Pts

For multi-object tracking, the basic idea is to detect multiple bounding boxes in current frame (that correspond to multiple cars) and match/track each of them to the appropriate bounding box in next frame.

2.2.1 Closest Bounding Box

Here once you have detected bounding boxes for each of the car in the image. You can do the following steps

- Filter out any of the false positives in the detection.
- For each bounding box in current frame find the closest bounding box in the previous frame.
    1. Care should be taken such that the closest match (in previous frame) for a bounding box (in current frame) should not be *too far away* from each other, which can be achieved by thresholding the maximum possible shift in centroid in successive frames.
    2. You can also compare bounding box properties to check if you found a correct match. It could be that the ratio of areas of bounding box (in current frame) to it's match (in previous frame) should not be too far away from 1.
- You might also have *new* detections or *lost* detections in current frame. Again you can detect such results using the location of centroid as well as *thresholding* the distance between centroids of bounding box (in current frame) and it's match (in previous frame).

2.2.2 Using KLT Tracker

Here we use *pointTracker* function in MATLAB's Computer Vision Toolbox. You can follow the below guidelines while using a tracker.

- For each bounding box in previous image *extract feature points* and *track* those points in the current frame.
- Now that you have *source points* (from previous frame) matched to *destination points* (from current frame), find the *geometric transform* from *source* to *destination.*
- Apply the *geometric transform* for the bounding box (in previous frame) to get the updated bounding box for current frame.
- Repeat this for each bounding box in previous frame to get the updated bounding box in current frame.
- You can detect vehicles in one frame followed by just tracking (use above steps) in N subsequent frames, repeat this process. Care should be taken that N is not large enough so that you do not miss new detections in current frame.

You can use any tracking algorithm of your choice, but YOU SHOULD NOT USE MultiTracker function in MATLAB.

You need to label each bounding box with a tag (could be a number/letter/color of the bounding box etc.) which should be consistent across frames demonstrating that the car is tracked and not just detected.

YOU CAN USE ANY OTHER APPROACH as you feel comfortable, which might not be given here. In any case, PLEASE SUBMIT A REPORT explaining your approach with relevant outputs after each step.

3 Extra Credit

Achieve good results on `challenge.avi` video (atleast 25 seconds at 30 fps) - **upto 40 Pts**.

4 Submission Guidelines

Your submission **SHOULD be a ZIP folder** (no other file extensions) with the naming convention `YourDirectoryID_proj3.zip` on to ELMS/Canvas. Additionally follow the guidelines given below:

1. You will have a parent directory `P3_Submission`.
2. Under `P3_Submission/CarTracking` you will have three sub-folders `code`, `input` and `output`.
3. You should also submit a report (`Report.pdf`) under `P3_Submission/CarTracking` folder.

5 Useful Resources

- Understanding Haar Cascade classifier - Viola-Jones face detector. You can watch a video lecture cited in the same link for in-depth understanding.
- PointTracker in MATLAB and Face Tracking example.
- Cascade Classifier in OpenCV.
- For complete understanding on Optical Flow, KLT/MeanShift Trackers watch these excellent video lectures.

6 Acknowledgement

Dataset used is by courtesy of Udacity's *Self Driving Program* and *LISA* dataset.

7 Collaboration Policy

You are allowed to discuss the ideas with fellow students, but you need to give credits in the report. But the code you turn-in should be your own and if you **DO USE** (try not to and it is not permitted) other external codes - do cite them and you might get partial credits. For other honor code refer to the University of Maryland Honor Pledge.

DISCLAIMER: *Details provided in the project's pipeline might not be fully complete, but in any case we might keep adding additional information as we find something relevant. You should take the effort to search online for any help regarding function usage, however, any concept related queries can be discussed in TA Office Hours.*