Vehicle Detection and Tracking

Marcos Ahuizotl "Ahui" Fragoso Iñiguez March 6, 2017

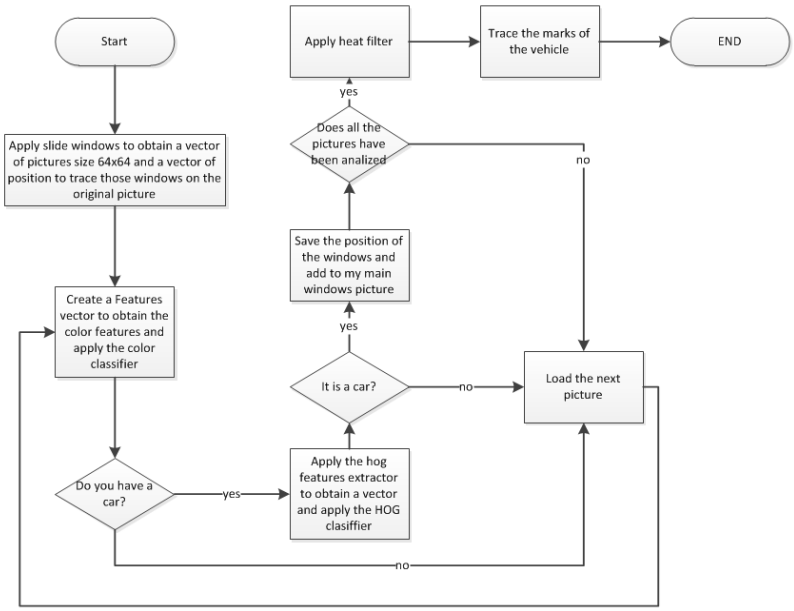
Goals

The goals / steps of this project are the following:

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
- Optionally, you can also apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector.
- Note: for those first two steps don't forget to normalize your features and randomize a selection for training and testing.
- Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
- Run your pipeline on a video stream (start with the test_video.mp4 and later implement on full project_video.mp4) and create a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles.
- Estimate a bounding box for vehicles detected.

Pipeline (single images)

The way I planned to aboard the project was the next one, the reason for me to apply this logic was that even when I tried the combined model it doesn´t work as good as I expected, so I decide to divide my classification in two parts, the first one was to check if there are chances to have a car in the section with the color classifier, then to confirm if it was a car I applied my HOG classifier, that way I don't have to do that many HOG transformations and I can run faster.



To do my classifiers I did two programs to obtain two separated classifiers, both have the way I loaded the data as similitude, what I did was to obtain all the data in the folders for cars and not cars, then I reduced my sample size to 2000 to avoid overfitting, and I applied a randomizer, so I can apply my feature extraction.

```
##*******************************************************
#Step 1: load all the required pictures for the trainning
cars = glob.glob('vehicles/*/*.png')
print("total of car pictures: ",len(cars))

notcars = glob.glob('non-vehicles/*/*.png')
print("total of non vehicles pictures: ",len(notcars))

testCar = random.randint(0, len(cars))
testNoCar = random.randint(0, len(notcars))

# Reduce the sample size because HOG features are slow to compute
# The quiz evaluator times out after 13s of CPU time
sample_size = 2000
cars = cars[0:sample_size]
notcars = notcars[0:sample_size]
print("total of car pictures: ",len(cars))
print("total of non vehicles pictures: ",len(notcars))
spatial = 32
histbin = 32
colorspace = 'YCrCb'# Can be RGB, HSV, LUV, HLS, YUV

cars, notcars = shuffle(cars,notcars)

car_features = extract_features(cars, colorspace, spatial_size=(spatial, spatial),
                                hist_bins=histbin)
notcar_features = extract_features(notcars, colorspace, spatial_size=(spatial, spatial),
                                   hist_bins=histbin)
```

Then I applied some operations to make my program to create some vectors which I can use to apply the SVC operations for my classifiers. A very important point is that I applied a 'rbf' kernel for my SVCs, because Linear kernel only give me as much as 93%, and with this kernel I get 99% for both cases.

I choose the nexr parameters for my transformations, first, the color space was YCrCb because after several experiments was the intermediate color space, because RGB gives me about 97% of accuracy and HLS gives me 100 accuracy, so the YcrCb give me 98% accuracy which makes me feel confident about the over fitting.

```
#*****************
spatial = 32
histbin = 32
colorspace = 'YCrCb'# Can be RGB, HSV, LUV, HLS, YUV
#******HOG********
orient = 9
pix_per_cell = 8
cell_per_block = 8
hog_channel = "ALL" # Can be 0, 1, 2, or "ALL"

#Picke Data
calibrator = "calibrationFiles.p"
with open(calibrator, mode='rb') as f:
    calibra = pickle.load(f)

mtx, dist = calibra['mtx'],calibra['dist']
```

Then my spatial and hist bin size are 32 because are also the intermediate case, when the speed and accuracy meets, and something you might notice is that I picked up 8 cells per block, my classifiers wasn't very good hence giving more blocks allows me to have more precision.

Finally I applied this information into a pipeline, you can find this pipeline in the "finalVehicleFinder" file, also for my HOG and Color classifiers you can check the file "hogTransformationTrainner" and "colorTrainner".

.

Pipeline (Video)

I only added some lines for the imagio so I can save my processing for each picture and transform that information into a video. This code is "finalVideoFinder"

Discussion

It was really complicated to develop this project because even when I applied my classifiers in several ways, it was never enough for the project, so I was thinking that maybe the cassifiers aren't the best choice, and I can use LeNet for example, and maybe could be faster.

Also I followed all the instructions from the Ryan Video and I never get the same performance of pictures than him, I don't know what I was doing wrong, so I uploaded the best performance I could get, maybe if it's not enough you can help me with directions.

Thank you.