Homework -3

COSC 6373 Computer Vision – Spring 2018

University Of Houston



Priscilla Imandi

PS ID : 1619570

**INTRODUCTION**

**Computer Vision**

Computer Vision is the science that gives ability to a machine to analyze and understand useful information from a single image or a sequence of images.

**Parking Lot Vacancy Detector**

Here, the problem statement is to find out how many cars are in the parking lot or more conveniently – how many empty parking spots are available in a parking lot. Generally, in huge cities the problem of parking has become a major concern and people end up wasting a lot of precious time in doing things like finding a spot to park. But, this problem can be solved by using object detection and detecting the number of cars already parked.

**Dataset**

The data can be gathered from the CCTV footage or google street view dynamically to give continuous updated results.

Usually, most of the parking lots are open and a wide range of climatic and weather conditions effect the view of the parking lots. If the parking lots are closed (in buildings) then a problem of illumination can also happen. In this homework, we will consider test and training images of cars from open lots in three different weather conditions – sunny, rainy and cloudy.

**Steps To be Followed**

The objective of the process is to find the number of empty parking spots in any given image.

**Step 1:**

We have a huge dataset – training is done on the images from the ‘sunny’ dataset. Since there are images from 3 different parking lots, we will consider positives from all the three samples. A mixture of data representing all the 3 lots at different times of the day can be taken.

On parsing the ground-truth .xml file – we will get to know where the cars are actually parked and we can extract these images of the cars to form the positive dataset.

The negative dataset can be anything apart from the cars.

**Step 2:**

Now, from these positive samples we have to create a positive vector file which is the required input format for the classifier. It is a .vec file which holds the data about the location and size of the positive images. It can be done using opencv\_createsamples.

**Step 3:**

Now, training the classifier using the negatives and the positive.vec file and specifying the feature type – either HAAR or LBP. It is done by using opencv\_traincascade. This generates a .xml file which is our classifier.

**Step 4:**

Now, we can use this cascade file(.xml) to detect the number of cars for an input image.

**IMPLEMENTATION**

1. **Training Set**

Using sunny scenario, images are extracted.

An xml parser – minidom.parse is used to extract the data from the tags.

We extract all the four contours of a spot. Now this is a bounding box which represents the car along its outline – which might be oriented along an angle according to the placement of the car. So consider the minRectArea using the function which gives a rectangular box that comprises of the oriented car area.

Now, this image is considered as a positive example.

Below are a few examples of positive images.

**   **

A small code snippet showing how an input which takes the four corners and generates an upright rectangle.

rectangle = cv2.minAreaRect(points)

A small code snippet showing how an input which takes the four corners and generates an upright rectangle.

Below are a few examples of negative samples

1. **Training Cascade Classifier**

Now, we need to create 2 text files one for positives samples and one for negative samples.

Classifier

Opencv\_createsamples

Positive.vec

Positive.txt

Negative.txt

A general pipeline on how to give input to a classifier

Here, the positive.txt is of the below format

Each line consists of the following data:

<location of image> <number of samples> <location -x,y,w,h of 1st positive sample> <location -x,y,w,h of 2nd positive sample> and so on.

The individual data can be separated by a space or a tab.

The negative.txt file is of the below format

Each line consists of the location of the negative sample

<location of image>

Generation of the positive vector file.



Here, positive24.vec is the output vector that createsamples generates from posdat.txt.

**Number of samples considered:**

Usually changing the number of samples did not make a huge difference on the how well the classifier could detect.

But the **quality of the samples** is very important – the better the quality of samples, the better the classifier can detect.

Initially I started with a dataset of 3000 positive and 2800 negative samples and obtained a decent classified even though there were many false positives.

Then I increased the classifier sample size to 6000 positives and 10000 positives and equal number of negatives, a slight improvement was seen in my classifier. But, the samples were not of high quality, i.e, even though the positives had only cars in them, the negative samples had parts of cars from the neighboring parking spots. And I considered only the data from the parking lots.

Then I improved the quality of my dataset by taking random pictures of trees, chairs, billboards – anything that is not a car and manually went through my negative dataset and eliminated the possibility of any occurrence of a car, and removed pictures that had cars or parts of cars.

Now I trained it on datasets of 2000 samples and 1800 samples with multiple stages. Now I observed that the detection has increased greatly.

**Number of Stages considered:**

To further increase the efficiency of the classifier, I increased the number of stages from 10 to 20 to 40. With increase in stages the classifier rectified its mistakes and learnt from them. But if I further increase the number of stages, there might be a possibility of overfitting.

But for the increase from 10 to 40 – there was an increase in efficiency.

Features Used – HAAR and LBP:

For stages 10 the efficiency of HAAR is higher – probably because all the calculations done by HAAR are in float whereas in int in LBP making HAAR achieve better precision. HAAR also took a relatively longer time to train when compared to LBP.

But, I observed that for 40 stages, LBP detects better than HAAR

Training for LBP and HAAR Features:

The below command can be used for training for LBP features

-data denotes where we need to place the cascade.xml file which is the classifier.

-bg denotes the negative.txt which contains data about negative samples.

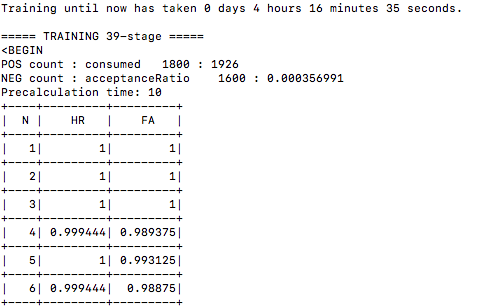
-numPos, -numNeg, -numStages denotes the number of positive samples, negative samples and number of stages

-minHitRate the number of hits per total detections

-w and -h can be taken as 24 which is the dimensions of the sample image.



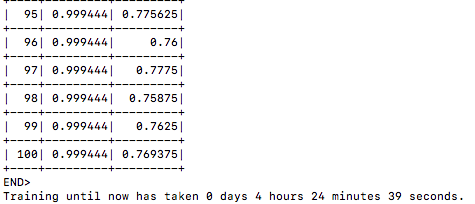
Below is a sample of how the training Hit Rate and False Alarm looks for LBP classifier at the 40th stage.



.

.

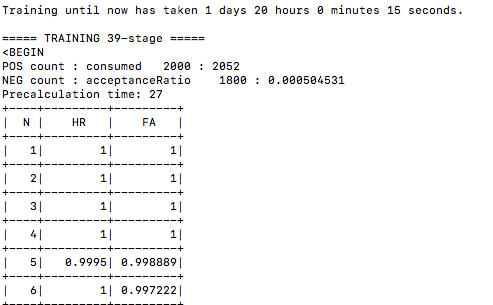
.



The below command is used to train for HAAR features

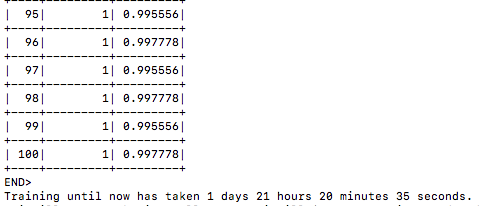


Below is a sample of how the training Hit Rate and False Alarm looks for HAAR classifier at the 40th stage.

****

**.**

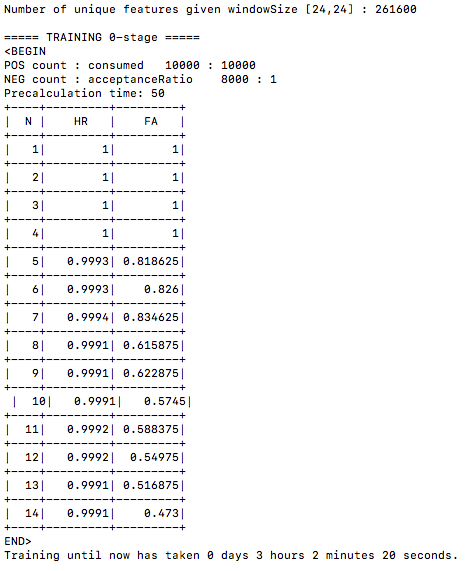
**.**

****

**Observation:**

1. From the above data it can be seen that LBP takes lesser time to train when compared to HAAR – due to the intensive float calculations done by HAAR.
2. The hitrate of HAAR is higher – with a hitrate of 1- means that almost all the detections are done perfectly and this is increasing as N increases.
3. The false alarm rate is also high indicating that a number of inaccurate predictions are being done.
4. Even though LBP doesn’t reach the perfect ‘1’ hitrate , the false alarm rate is lower indicating that it is more efficient than HAAR.

The below image is another depiction of HAAR.



From the above image, we can see that for HAAR the hit rate is nearly 1 and False alarm rate is considerably low when compared to LBP.

1. **Car Detection**

Images from cloudy and rainy have to be taken to as testing images. Generally for any learning process – the dataset is divided into testing and training(and validation sometimes). Here, our test images are not as good as the training images as they contain noise( rain and cloudy) – so the accuracy is not as high as the sunny images.

Detection is done using

cars = car\_cascade.detectMultiScale(image, 3, 5)

Here, car\_cascade is the classifier object, the parameters are image which is to be tested, scale and min neighbors.

Scale is the size of the detection, which should always be greater than 1.

Min neighbors is used to eliminate false positives, if we give 0 we get a lot of detections. This occurs because the classifier uses a sliding window and then it resizes the window and slides again until resizing cannot happen anymore. Because of sliding it multiple times, multiple detections happen at the same point. So a neighborhood approach helps where the box is passed if it is in the neighborhood of other boxes. This factor determines how much neighborhood is required to pass it as a detection.

The higher, the lesser false positives, but very high values result in the loss of true positives as well.

1. **Parking Lot Analysis**

The below table has test images from rainy(26-50) and cloudy(1-25)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Image** | **Lbp/ Haar** | **Stages** | **No of Positive** | **No of Negatives** | **TP** | **FP** | **Accuracy** |
| **Img 1** | **H**aar | 40 | 2000 | 1800 | 35 | 5 | 87.5 |
| **Img 2** | **H**aar | 40 | 2000 | 1800 | 31 | 15 | 79 |
| **Img 3** | **Lbp** | 40 | 2000 | 1800 | 35 | 14 | 87.5 |
| **Img 4** | **Lbp** | 40 | 2000 | 1800 | 40 | 7 | 100 |
| **Img 5** | **Lbp** | 40 | 2000 | 1800 | 4 | 2 | 100 |
| **Img 6** | **H**aar | 10 | 1800 | 1600 | 12 | 25 | 50.2 |
| **Img 7** | **lbp** | 10 | 1800 | 1600 | 31 | 34 | 73.5 |
| **Img 8** | **H**aar | 20 | 2000 | 1800 | 0 | 1 | 0 |
| **Img 9** | **H**aar | 10 | 1800 | 1600 | 2 | 4 | 66.6 |
| **Img10** | **H**aar | 40 | 2000 | 1800 | 101 | 20 | 91.2 |
| **Img11** | **Lbp** | 40 | 2000 | 1800 | 59 | 7 | 93.6 |
| **Img12** | **Haar** | 40 | 2000 | 1800 | 53 | 3 | 84 |
| **Img13** | **H**aar | 20 | 2000 | 1800 | 65 | 12 | 69.1 |
| **Img14** | **Lbp** | 20 | 2000 | 1800 | 73 | 10 | 78 |
| **Img15** | **Lbp** | 10 | 2000 | 1800 | 65 | 12 | 69.8 |
| **Img16** | **H**aar | 10 | 2000 | 1800 | 42 | 19 | 32 |
| **Img17** | **Lbp** | 40 | 2000 | 1800 | 61 | 3 | 83 |
| **Img18** | **H**aar | 40 | 2000 | 1800 | 59 | 7 | 76 |
| **Img19** | **H**aar | 10 | 2000 | 1800 | 36 | 12 | 66 |
| **Img20** | **Lbp** | 20 | 2000 | 1800 | 62 | 5 | 80 |
| **Img21** | **H**aar | 40 | 2000 | 1800 | 70 | 9 | 79 |
| **Img22** | **Lbp** | 40 | 2000 | 1800 | 72 | 8 | 82 |
| **Img23** | **H**aar | 10 | 1800 | 1600 | 42 | 98 | 16 |
| **Img24** | **Lbp** | 10 | 1800 | 1600 | 12 | 67 | 30 |
| **Img25** | **H**aar | 20 | 2000 | 1800 | 11 | 9 | 39 |
| **Img26** | **Lbp** | 20 | 2000 | 1800 | 14 | 13 | 50 |
| **Img27** | **Lbp** | 10 | 1800 | 1600 | 11 | 9 | 39 |
| **Img28** | **Lbp** | 10 | 1800 | 1600 | 10 | 12 | 35.7 |
| **Img29** | **H**aar | 10 | 1800 | 1600 | 38 | 24 | 19.6 |
| **Img30** | **H**aar | 10 | 1800 | 1600 | 3 | 14 | 30 |
| **Img31** | **H**aar | 40 | 2000 | 1800 | 10 | 1 | 81 |
| **Img32** | **Lbp** | 40 | 2000 | 1800 | 14 | 1 | 87 |
| **Img33** | **Lbp** | 20 | 2000 | 1800 | 16 | 10 | 57.1 |
| **Img34** | **H**aar | 20 | 2000 | 1800 | 11 | 8 | 39.2 |
| **Img35** | **Lbp** | 20 | 2000 | 1800 | 61 | 5 | 74.4 |
| **Img36** | **H**aar | 40 | 2000 | 1800 | 60 | 7 | 73.1 |
| **Img37** | **Haar** | 20 | 2000 | 1800 | 48 | 15 | 53.2 |
| **Img38** | **H**aar | 10 | 1800 | 1600 | 18 | 54 | 22 |
| **Img39** | **H**aar | 10 | 1800 | 1600 | 19 | 58 | 12 |
| **Img40** | **H**aar | 40 | 2000 | 1800 | 67 | 6 | 72 |
| **Img41** | **Lbp** | 40 | 2000 | 1800 | 60 | 6 | 64.5 |
| **Img42** | **Lbp** | 40 | 2000 | 1800 | 83 | 11 | 89.2 |
| **Img43** | **H**aar | 40 | 2000 | 1800 | 79 | 6 | 84.9 |
| **Img44** | **Haar** | 40 | 2000 | 1800 | 85 | 9 | 90.4 |
| **Img45** | **Lbp** | 40 | 2000 | 1800 | 99 | 8 | 97.2 |
| **Img47** | **Lbp** | 20 | 2000 | 1800 | 85 | 9 | 90.4 |
| **Img48** | **H**aar | 20 | 2000 | 1800 | 81 | 15 | 86.5 |
| **Img49** | **Haar** | 10 | 1800 | 1600 | 63 | 37 | 53.2 |
| **Img50** | **Lbp** | 10 | 1800 | 1600 | 72 | 41 | 60.6 |

**Points To Note – Thoughts Of Inference**

1. While generating the positive samples, if we use the cars along their contours, then the classifier is not efficient as it is loosing very important data – the outlines of the car. Even, though there is some background it is better to consider that rather than loosing important features(outlines) of the car.
2. For different parking lots different data can be considered, and individual classifiers can be trained. – This gives a better detection because the quality of the positives will be higher with respect to that one parking lot.

I have considered all three lots together, so I have chosen samples from ‘sunny’ folders of different days at different times of the day – so that cars illuminated differently will be considered, and any weather changes also will be captured in the data. These positive samples are hand-picked for better quality.

1. The different lots have different issues – cloudy may have the problem of illumination as the pixels in the cloudy data will always be at a lower intensity when compared to the images in the sunny data.

The rainy images might be occluded a little because of the occurance of rain. Or the cars themselves may have a layer of water – distorting the features a little.

The sunny data might have the problem of shadows cast by the cars as the sun rises and sets.

1. Here, an important point to note is that not all the cars are given in the ground-truth file, i.e, there are some cars in the image whose location is not mentioned in the file, making it a false positive.

In the below image, the green boxes are the detected boxes whereas the purple boxes are the boxes from the .xml file. If we notice the row of cars in the middle is not given in the .xml file but our classifier detects them. But this would in turn decrease the accuracy of the classifier since we are comparing it with the ground truth file.



1. Sometimes, there might be multiple detections at the same point, in such cases even though it is a correct detection, it has to be considered only once. Below is an image of a detection that spans over multiple cars – though it cannot be considered.

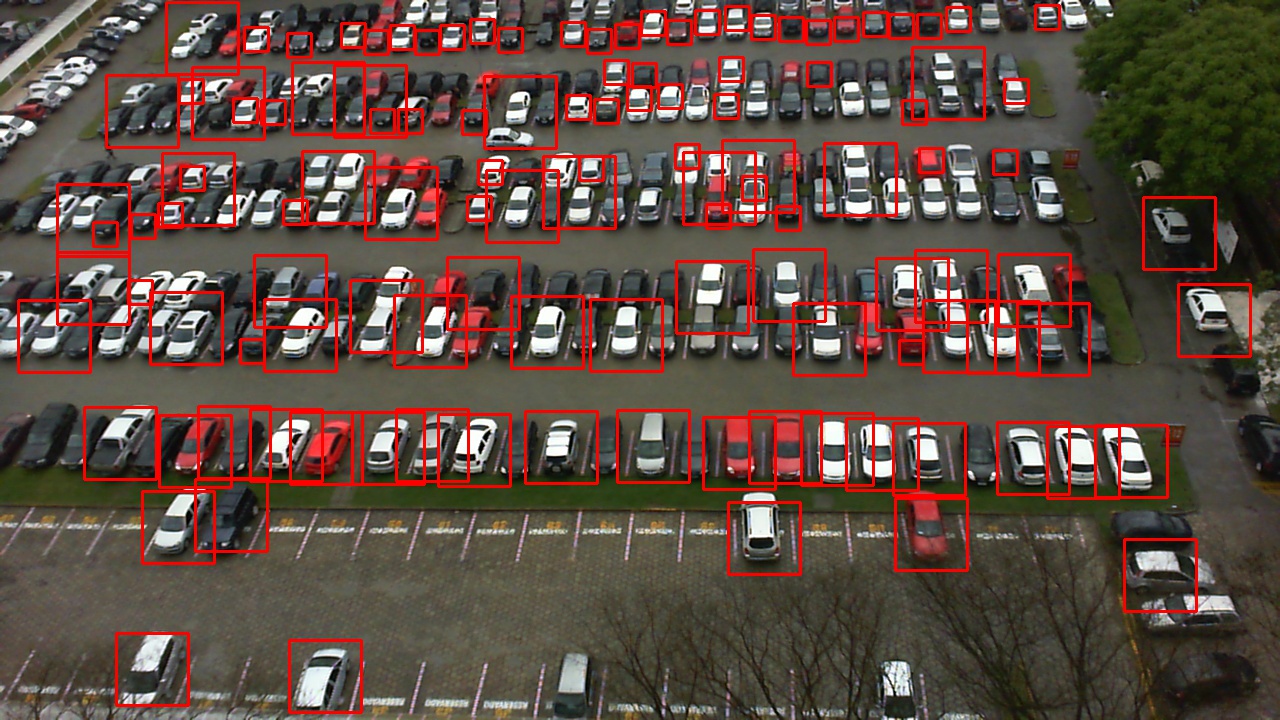


1. Parameters of the Detect MultiScale as discussed above(in 3.Car Detection) are very important. For the same image, with different values for the parameters, we obtain different results. The same classifier detects differently for different parameter values. The parameters are tuned for optimal results.

For scale = 7, minneighbors = 15, there are negligible detections.



For scale = 3, minneighbors = 5, there are multiple detections



1. Size of the sliding window, while generating the vector file it is very important to keep the size of the window in mind, as this has to be maintained even in the training. The dimensions of the samples should be nearly equal to the window size that we choose so that the classifier detects better.
2. Intersection Over Union denotes the Area of Overlap/ Area of Union. That is, if we consider two bounding boxes one from the ground-truth file and the other from the classifier. Then we compute the area under the boxes. Area of Overlap is the area common to both the boxes and area of union is the total area under both the boxes.

The IOU values are ranging over wide range – if the detections do not match at all, then the values are negatives and we can discard that comparison.

If the values are positive then there is overlap. Optimal IOU value ranges from 20-40 after testing for various images

1. Another important observation is that if there are more cars in a lot then we get higher accuracy, than an empty lot. This is due to detection of False positives.

If no True Positives are detected then we consider it as 0.

1. Sample detection

