

Haar Cascades

Saurabh Ghanekar





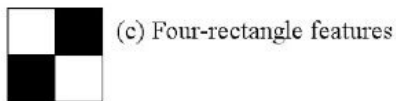
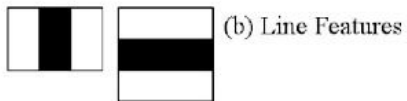
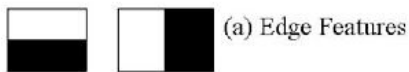
What is Haar Cascade?

- Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in 2001.
- It is a machine learning based approach where a cascade function (the **cascade** algorithm is a numerical method for calculating **function** values of the basic scaling and wavelet **functions** of a discrete wavelet transform using an iterative algorithm) is trained from a lot of positive and negative images. It is then used to detect objects in other images.



How does it work?

- The algorithm needs a lot of positive images (images of object to detect) and negative images (images without the object to detect) to train the classifier. Then we need to extract features from it.
- For this, Haar features are used. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.



$$f_i = \text{Sum}(r_{i,white}) - \text{Sum}(r_{i,black})$$

$$h_i(x) = \begin{cases} s_i & \text{if } f_i > \theta_i \\ -s_i & \text{if } f_i < \theta_i \end{cases}$$



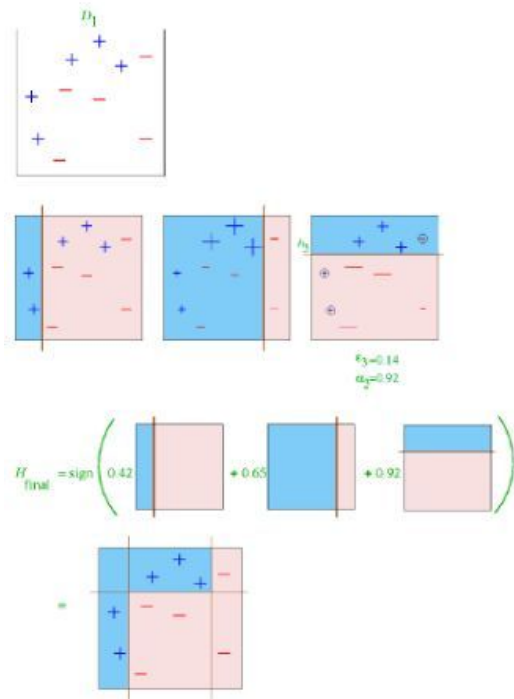
How does it work?

- The object detection framework employs a variant of the learning algorithm AdaBoost to both select and to train classifiers that use them. This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers.

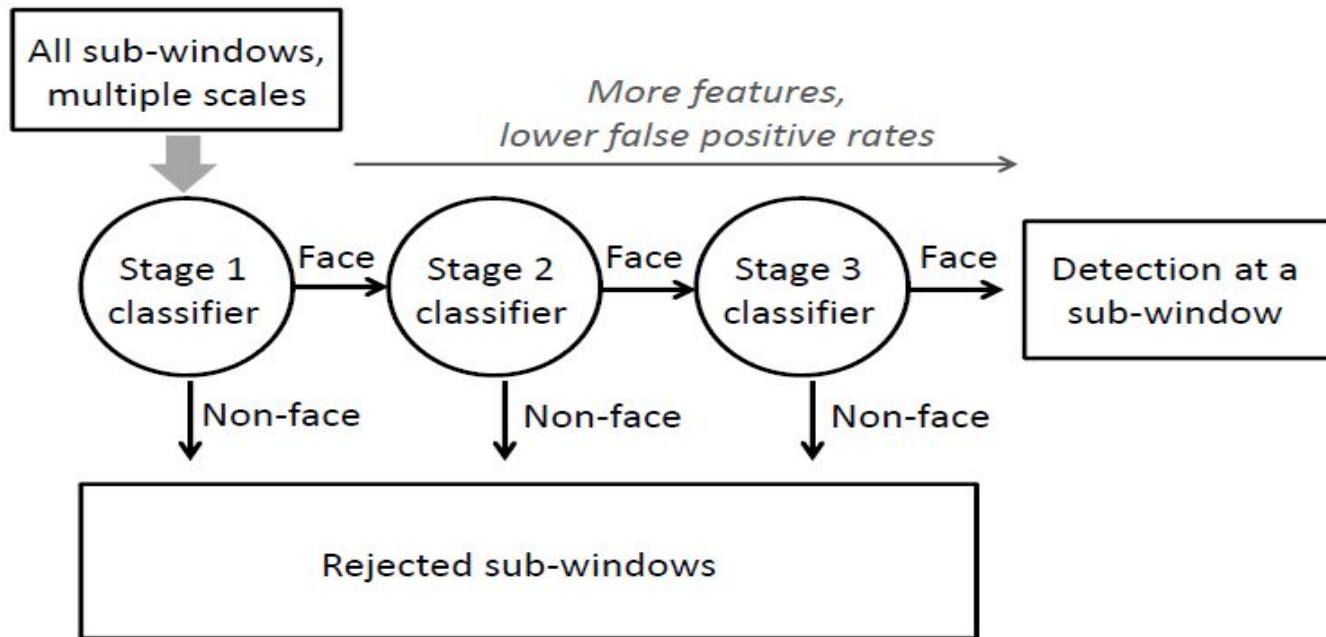
$$h(\mathbf{x}) = \text{sgn}\left(\sum_{j=1}^M \alpha_j h_j(\mathbf{x})\right)$$

- Each weak classifier is a threshold function based on the feature f_j

$$h_j(\mathbf{x}) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases}$$



How does it work?





Creating a custom Haar Cascade

- First I downloaded and resized the negative dataset using the following script:

I used one positive image as I decided to use my negative dataset to create my positive dataset by superimposing the positive image on the negative images.



stop.jpg (My Positive Image)

```
download_images.py x
1 import urllib.request
2 import cv2
3 import numpy as np
4 import os
5
6 def store_raw_images():
7     neg_images_link = "http://www.image-net.org/api/text/imagenet.synset.geturls?wnid=n04096066"
8     #neg_images_link = "http://www.image-net.org/api/text/imagenet.synset.geturls?wnid=n03244388"
9     neg_image_urls = urllib.request.urlopen(neg_images_link).read().decode()
10    pic_num = 1
11
12    if not os.path.exists("neg"):
13        os.mkdir("neg")
14
15    for i in neg_image_urls.split('\n'):
16        try:
17            print(i + " - " + str(pic_num))
18            urllib.request.urlretrieve(i, "neg/" + str(pic_num) + ".jpg")
19            img = cv2.imread("neg/" + str(pic_num) + ".jpg")
20
21            resized_image = cv2.resize(img, (640,400))
22            cv2.imwrite("neg/" + str(pic_num) + ".jpg", resized_image)
23            pic_num += 1
24
25        except Exception as e:
26            print(str(e))
27
28    store_raw_images()
```



Creating a custom Haar Cascade

- Then I created a descriptor file for these negatives(downloaded approx. 2000 images) using the following script:

```
createPath.py x
1  import os
2  import cv2
3
4  def create_neg():
5      for file_type in ["positives", "negatives"]:
6          for img in os.listdir(file_type):
7              if file_type == "negatives":
8                  line = file_type+"/"+img+"\n"
9                  with open("bg.txt", "a") as f:
10                     f.write(line)
11              elif file_type == 'positives':
12                  line = img+' 1 0 0 50 50\n'
13                  with open('info.dat','a') as f:
14                     f.write(line)
15
16  create_neg()
```



Creating a custom Haar Cascade

- Once the descriptor file is done, I checked if I have all the necessary files to create my positive dataset. I ran the following command line in the terminal to create my positive dataset,

```
opencv_createsamples -img stop.jpg -bg bg.txt -info info/info.lst -pngoutput  
info -maxxangle 0.5 -maxyangle 0.5 -maxzangle 0.5 -num 1950
```



My output



Creating a custom Haar Cascade

- Now that I have my positive images, I created a vector file, which is basically where I stitch all of my positive images together. I used the following command to do get my output,

```
opencv_createsamples -info info/info.lst -num 1950 -w 20 -h 20 -vec positives.vec
```



Creating a custom Haar Cascade

- Once the “positive.vec” file is created, created a new folder, “data” where my output will be stored. I ran the following command line in the terminal to train and get my final Haar Cascade file,

```
opencv_traincascade -data data -vec positives.vec -bg bg.txt -numPos 1800 -numNeg 900  
-numStages 10 -w 20 -h 20
```

```
cascade.xml
1  <?xml version="1.0"?>
2  <opencv_storage>
3  <cascade>
4  <stageType>BOOST</stageType>
5  <featureType>HAAR</featureType>
6  <height>20</height>
7  <width>20</width>
8  <stageParams>
9  <boostType>GAB</boostType>
10 <minHitRate>9.9800002574920654e-01</minHitRate>
11 <maxFalseAlarm>5.000000000000000e-01</maxFalseAlarm>
12 <weightTrimRate>9.499999999999999e-01</weightTrimRate>
13 <maxDepth>1</maxDepth>
14 <maxWeakCount>100</maxWeakCount>
15 </stageParams>
16 <catCount>0</catCount>
17 <featSize>1</featSize>
18 <mode>BASIC</mode>
19 <stageNum>15</stageNum>
20 <stages>
21 <!-- stage 0 -->
22 <!--
23 <maxWeakCount>6</maxWeakCount>
24 <stageThreshold>1.1638636589050293e+00</stageThreshold>
25 <weakClassifiers>
26 <!--
27 <internalNodes>
28 <0 -1 5 8.170479908585484e-03</internalNodes>
29 <leafValues>
30 <-6.7494356632232666e-01 6.8061488866806030e-01</leafValues>
31 <!--
32 <internalNodes>
33 <0 -1 63 -4.5486129820346832e-03</internalNodes>
34 <leafValues>
35 <6.4769452810287476e-01 -5.8978581428527832e-01</leafValues>
36 <!--
```

My outputs

```
cascade.xml
1510 <!--
1511 <features>
1512 <!--
1513 <rects>
1514 <0 0 2 2 -1</rects>
1515 <!--
1516 <rects>
1517 <1 0 1 2 2</rects>
1518 <tilted-0</tilted>
1519 <!--
1520 <rects>
1521 <0 0 2 17 -1</rects>
1522 <!--
1523 <rects>
1524 <1 0 1 17 2</rects>
1525 <tilted-0</tilted>
1526 <!--
1527 <rects>
1528 <0 1 2 16 -1</rects>
1529 <!--
1530 <rects>
1531 <1 1 16 2</rects>
1532 <tilted-0</tilted>
1533 <!--
1534 <rects>
1535 <0 1 2 17 -1</rects>
1536 <!--
1537 <rects>
1538 <1 1 17 2</rects>
1539 <tilted-0</tilted>
1540 <!--
1541 <rects>
1542 <0 2 20 6 -1</rects>
1543 <!--
1544 <rects>
1545 <10 2 10 6 2</rects>
1546 <tilted-0</tilted>
```



Implementing our Haar Cascade in Python

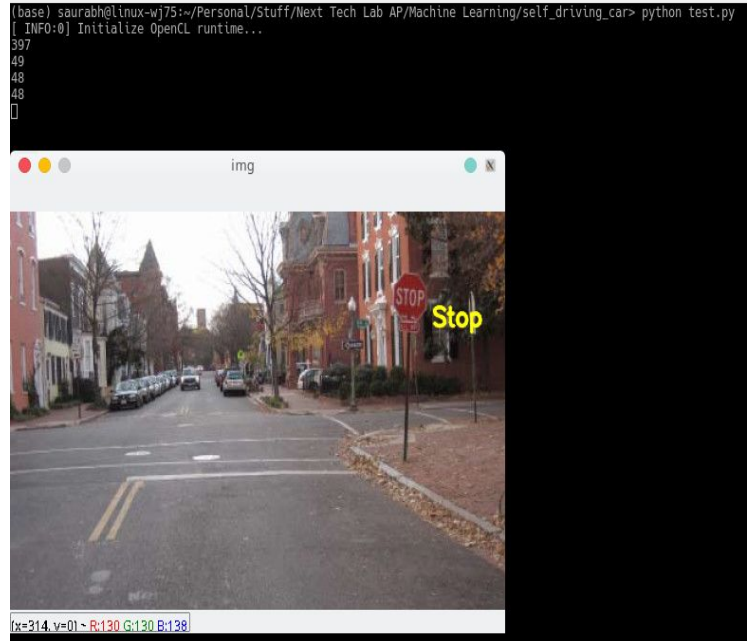
- Once my training was done, I used my Haar Cascade to detect stop signs within images. The code to use my Haar Cascade in python is as follows,

```
test.py x
1  import numpy as np
2  import cv2
3
4  stop_cascade = cv2.CascadeClassifier('cascade.xml')
5
6  img = cv2.imread('stop6.jpg')
7
8  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
9  stop = stop_cascade.detectMultiScale(gray, 1.1, 10)
10
11 for (x,y,w,h) in stop:
12     #cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
13     font = cv2.FONT_HERSHEY_SIMPLEX
14     cv2.putText(img,'Stop',(x+w,y+h), font, 0.75,(11,255,255),2,cv2.LINE_AA)
15     print(x)
16     print(y)
17     print(w)
18     print(h)
19 cv2.imshow('img',img)
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
22
```



Implementing our Haar Cascade in Python

- So as I executed my python code, I got the following results.





Any Questions?



@Saurabh98G



<https://www.linkedin.com/in/saurabhghanekar/>



<https://dexter2389.github.io/>