Haar Cascades

Saurabh Ghanekar

What is Haar Cascade?

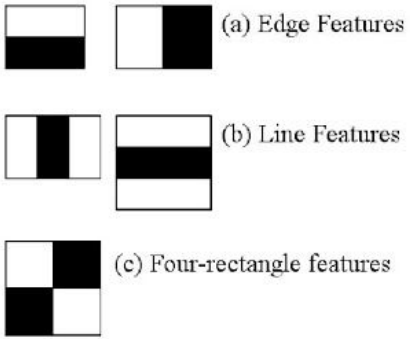
- Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in 2001.

- 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.

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



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

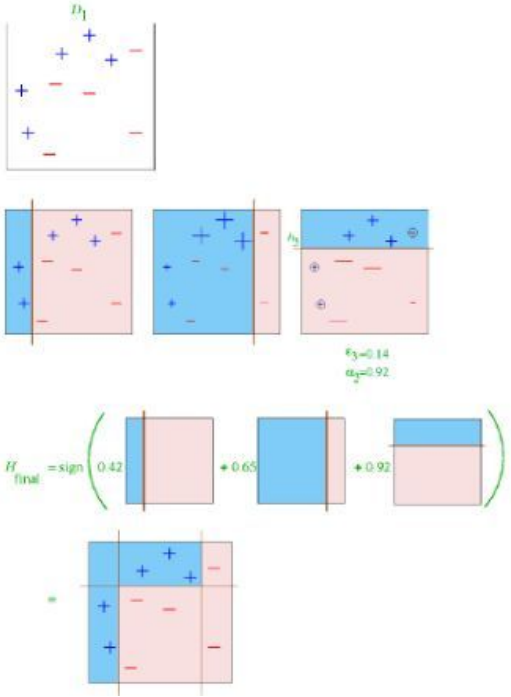
How does it work?

- 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.

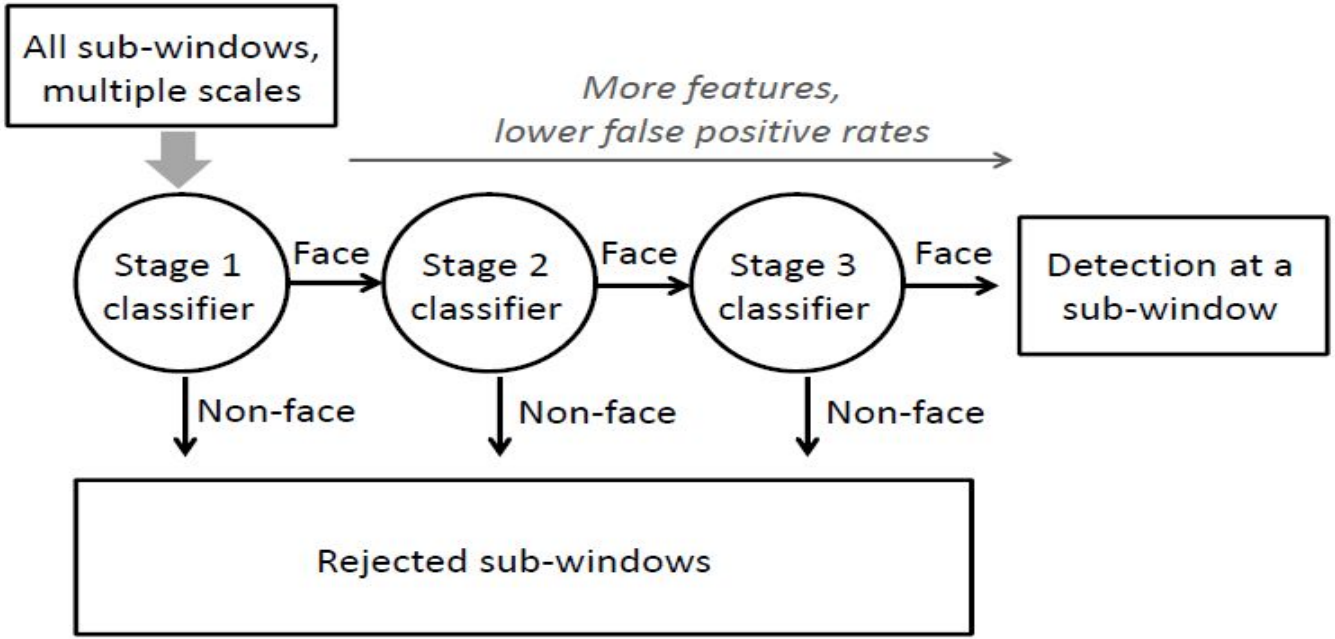
$$h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$$

- 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

```python
import urllib.request
import cv2
import numpy as np
import os

def store_raw_images():
    neg_images_link = "http://www.image-net.org/api/text/imagenet.synset.geturls?wnid=n04096066"
    #neg_images_link = "http://www.image-net.org/api/text/imagenet.synset.geturls?wnid=n03244388"
    neg_image_urls = urllib.request.urlopen(neg_images_link).read().decode()
    pic_num = 1

    if not os.path.exists("neg"):
        os.mkdir("neg")

    for i in neg_image_urls.split('\n'):
        try:
            print(i + "  -  " + str(pic_num))
            urllib.request.urlretrieve(i, "neg/" + str(pic_num) + ".jpg")
            img = cv2.imread("neg/" + str(pic_num) + ".jpg")

            resized_image = cv2.resize(img, (640,400))
            cv2.imwrite("neg/" + str(pic_num) + ".jpg", resized_image)
            pic_num += 1

        except Exception as e:
            print(str(e))

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

```python
import os
import cv2

def create_neg():
    for file_type in ["positives", "negatives"]:
        for img in os.listdir(file_type):
            if file_type == "negatives":
                line = file_type+"/"+img+"\n"
                with open("bg.txt", "a") as f:
                    f.write(line)
            elif file_type == 'positives':
                line = img+' 1 0 0 50 50\n'
                with open('info.dat','a') as f:
                    f.write(line)

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

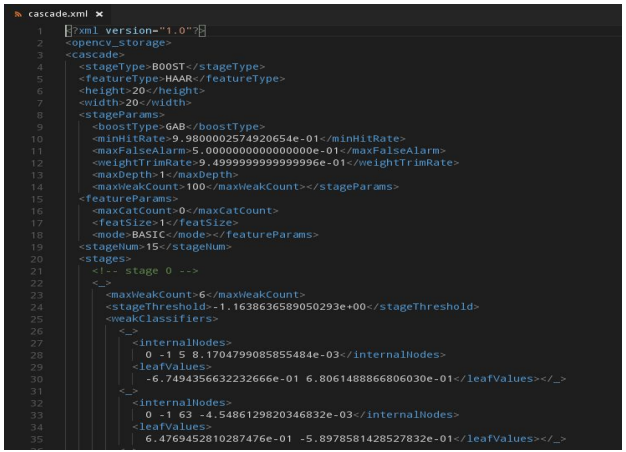
- Now that I have my positive images, I created a vector file, which is basically where I stitch all of my positive images together. I used the following command to do get my output,

```
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
```



My outputs

Implementing our Haar Cascade in Python

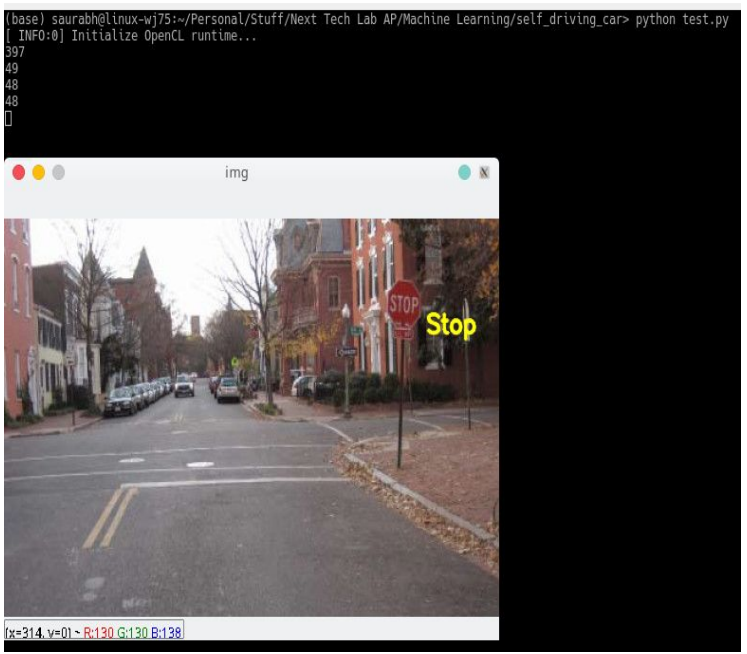
- Once my training was done, I used my Haar Cascade to detect stop signs within images. The code to use my Haar Cascade in python is as follows,

test.py

```python
import numpy as np
import cv2

stop_cascade = cv2.CascadeClassifier('cascade.xml')

img = cv2.imread('stop6.jpg')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
stop = stop_cascade.detectMultiScale(gray, 1.1, 10)

for (x,y,w,h) in stop:
    #cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.putText(img,'Stop',(x+w,y+h), font, 0.75,(11,255,255),2,cv2.LINE_AA)
    print(x)
    print(y)
    print(w)
    print(h)
cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

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/