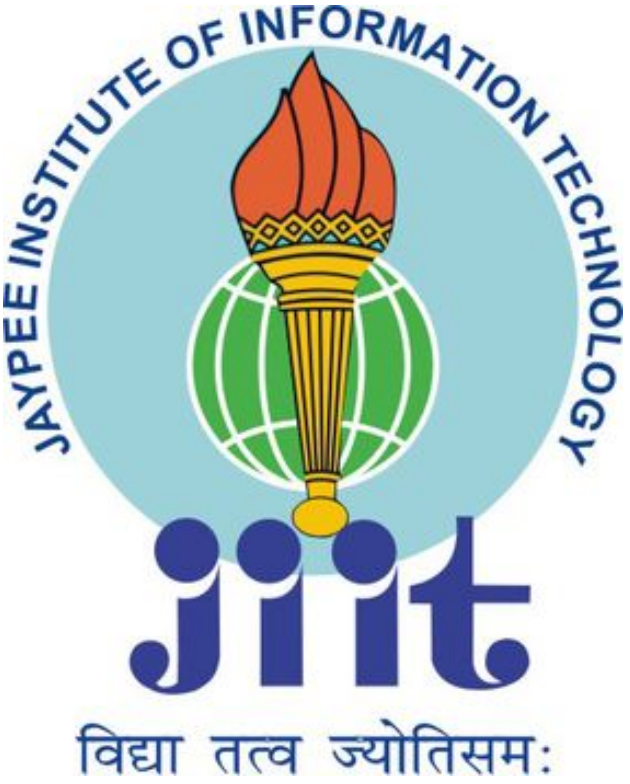
ALGORITHMS AND PROBLEM SOLVING LAB

PROJECT SYNOPSIS

Face Detection

(4TH SEMESTER 2019-2020)



Made By:

Sanchit Sahdev 17103128 **[B2]**

Akshay Gautam 17103010 **[B1]**

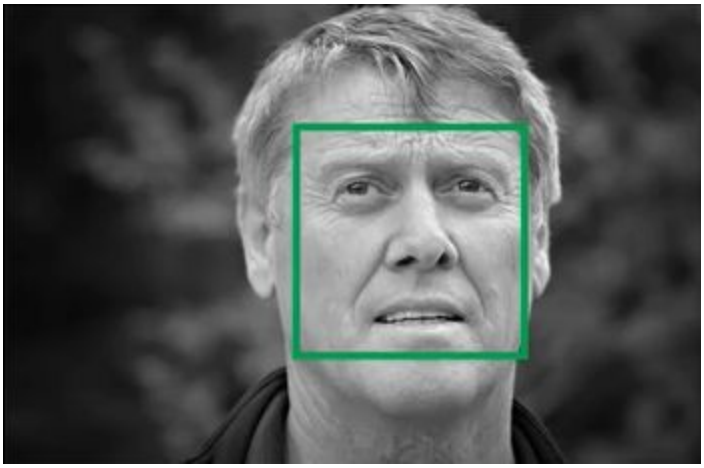
Ankit Sharma 17103031 **[B1]**

INTRODUCTION

Face Detection is a common application used in many cameras. You can notice that when you are about to take a picture a rectangular box appears around your head. This is called face detection, in the case of taking a picture, it is mostly used so as to automatically adjust the lighting so that you get the perfect picture.

THE VIOLA-JONES ALGORITHM

Viola-Jones object detection framework was an algorithm proposed in 2001 by Paul Viola and Michael Jones. It aimed to solve the problem of detecting faces in an image. While a human can do this easily, a computer needs precise instructions and constraints. To make the task more manageable, Viola–Jones requires full view frontal upright faces. Thus in order to be detected, the entire face must point towards the camera and should not be tilted to either side. While it seems these constraints could diminish the algorithm's utility somewhat, because the detection step is most often followed by a recognition step, in practice these limits on pose are quite acceptable.



IMPLEMENTATION OF THE ALGORITHM

To detect a human in an image, this algorithm is going to look at the most relevant features such as the eyes, nose, forehead, lips and eyebrows.

But how is it going to detect all these features?
It is going to use ***Haar Features.***

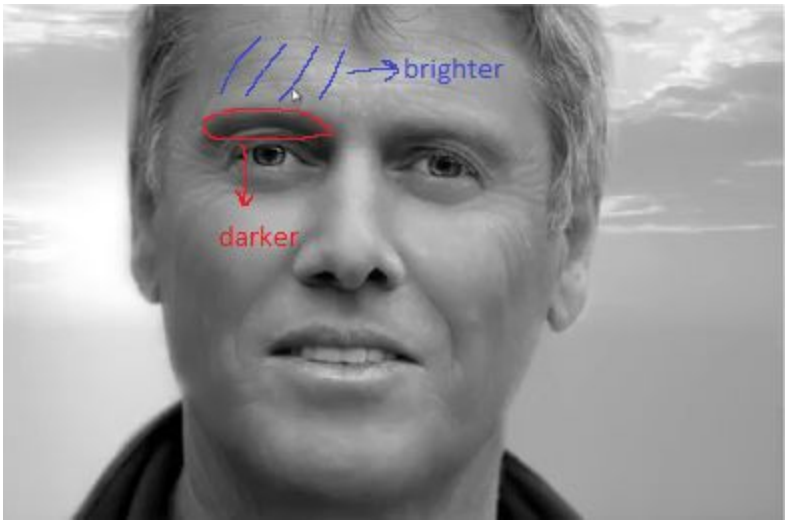
1. Understanding Haar Features

Haar features or Haar-wavelet are a sequence of rescaled rectangular shaped function. They were proposed by Hungarian mathematician Alfred Haar in 1909.

Therefore, Haar features are the relevant features that are going to be used in face detection.We are going to assign Haar features to every single feature on a human face. For Example- There are Edge Features and Line Features

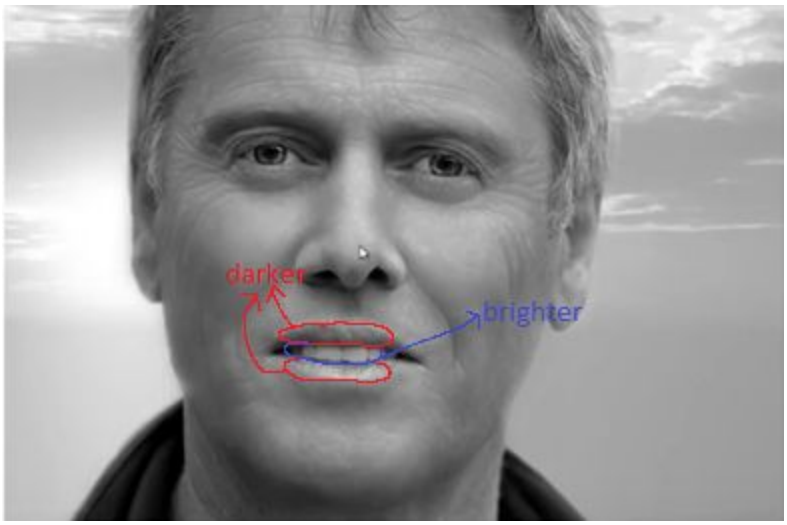
Edge features- As the name says, it can detect edges effectively.As you can see in the picture above, white and black pixels are used. But of course, in real images there aren't completely black or white pixels. However, this problem can be solved by converting the image to a grayscale picture. Now we can get close to these white/black pixels.

In a grayscale image, we will take the value of each pixel from 0 to 1.
0 being completely white and
1 being completely black.



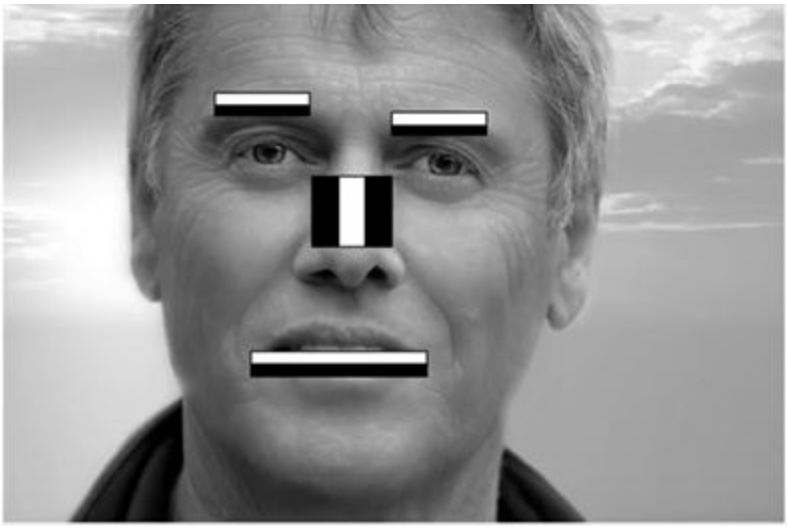
Now, understanding from the picture above, we can see that the area of the eyebrow (marked by Red) is significantly darker than are of the forehead (marked by Blue).
Using this major difference of the pixel values (Remember 0 being white and 1 being black) we can detect edges in pictures.

<u>Line Features-</u>These can detect lines.
Same as in edge features, we must convert the picture to grayscale so we can deal with the white and black pixels



As we can see can detect a line from the above image by separating the dark (marked by Red) and the bright pixels (marked by Blue).

We can also use this to detect nose (dark-bright-dark)
It can be understood by the picture below

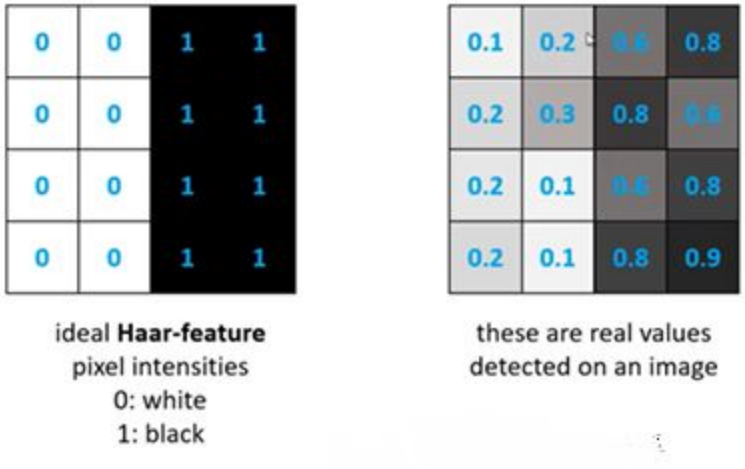


Hence we are able to assign the Haar features on the *most relevant* features of the human face. Now we can detect human faces based on the most relevant features detected.

Now further expanding on “pixel values”

As it was mentioned before, black pixels would be assigned the value 1 whereas white pictures would be assigned the value 0. By converting the images to grayscale it would be easier to deal with the input images. However it is not possible to have only the values 0 and 1 in an image. So these values will range between 0 to 1 based on their intensities.

To better understand this, refer to the picture below-



Now, the algorithm is going to compare how close the real scenario is to the ideal case.
To do that-

1. Sum up white pixel intensities and find the average.
2. Sum up black pixel intensities and find the average.
3. Find “D” where D is the difference of (2) and (1).

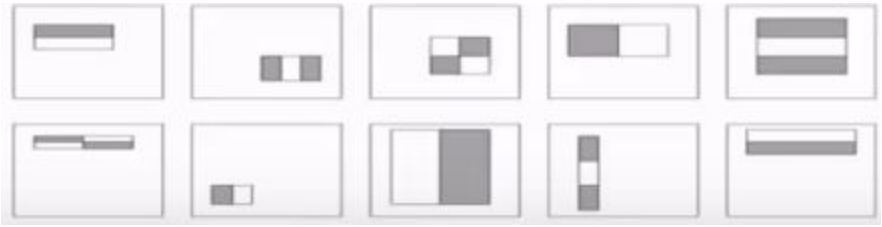
Performing the operations in the cases above, we get-

- **D for Ideal Case = Avg(dark pixels) - Avg(white pixels)**
  = (8/8) - (0/8)
  = 1

- **D for Real Case = 0.74 - 0.18**
  = 0.56

Now, the closer the value is to 1, the more likely it is that we have found a Haar Feature.

But here arises a problem. It would be extremely time consuming to calculate average and D value of many segments of the image as this algorithm using so many features.

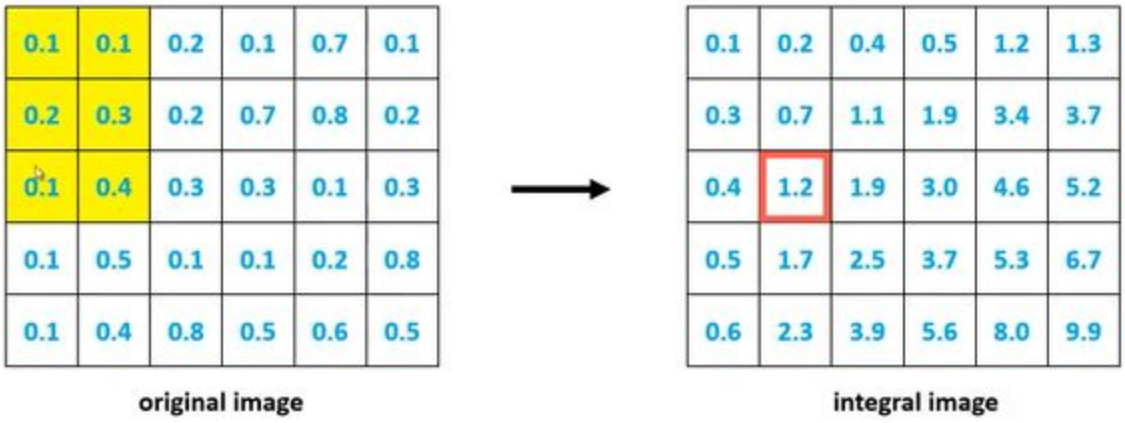


Here is where the Viola-Jones algorithm uses *Integral Images* to improve it's time complexity.

2. Optimizing the Algorithm with the help of Integral Images

We use Integral Images to reduce the running time to 0(1).

Now the in an integral image, the value of the pixels is determined by the sum of all the pixels from the left and above, like this-

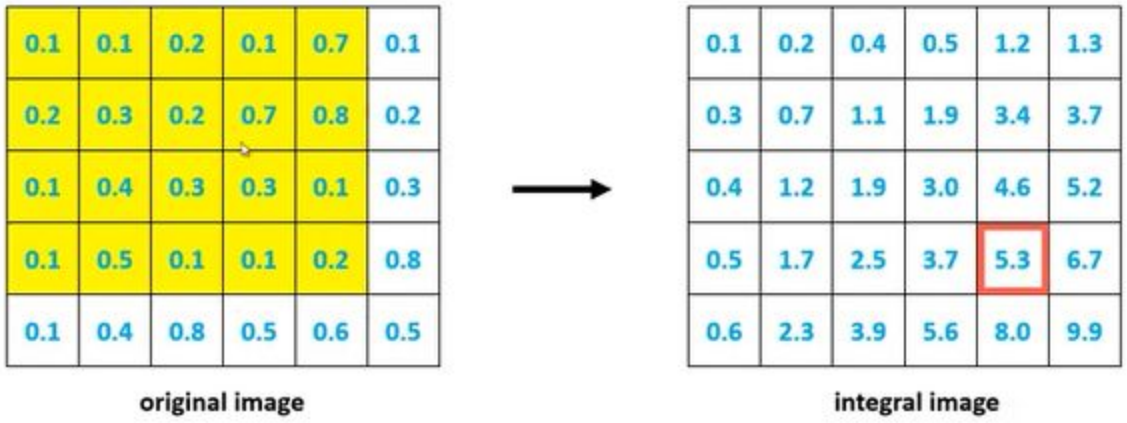


0.1	0.1	0.2	0.1	0.7	0.1
0.2	0.3	0.2	0.7	0.8	0.2
0.1	0.4	0.3	0.3	0.1	0.3
0.1	0.5	0.1	0.1	0.2	0.8
0.1	0.4	0.8	0.5	0.6	0.5

original image

0.1	0.2	0.4	0.5	1.2	1.3
0.3	0.7	1.1	1.9	3.4	3.7
0.4	1.2	1.9	3.0	4.6	5.2
0.5	1.7	2.5	3.7	5.3	6.7
0.6	2.3	3.9	5.6	8.0	9.9

integral image



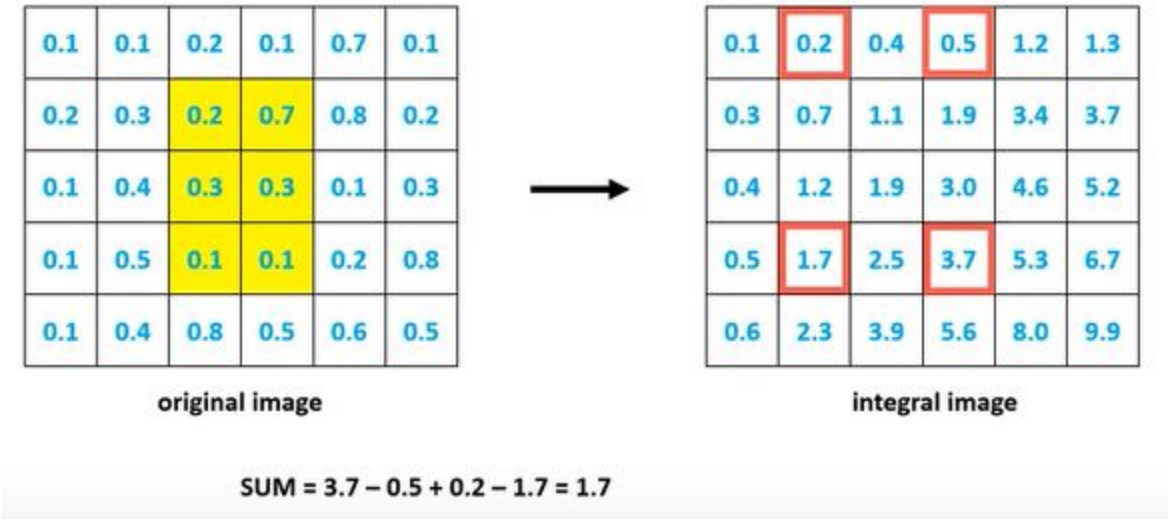
0.1	0.1	0.2	0.1	0.7	0.1
0.2	0.3	0.2	0.7	0.8	0.2
0.1	0.4	0.3	0.3	0.1	0.3
0.1	0.5	0.1	0.1	0.2	0.8
0.1	0.4	0.8	0.5	0.6	0.5

original image

0.1	0.2	0.4	0.5	1.2	1.3
0.3	0.7	1.1	1.9	3.4	3.7
0.4	1.2	1.9	3.0	4.6	5.2
0.5	1.7	2.5	3.7	5.3	6.7
0.6	2.3	3.9	5.6	8.0	9.9

integral image

For finding the value of a particular region-



As we can see, instead of dealing with all the pixels of that area, we are only using the corners to calculate the area.

3. Optimizing the Algorithm with the help of Adaboost

As mentioned previously, the algorithm has to take in many Haar features and its value and compare it to the image. The number of features could easily reach above 100,000 in an image. Here is where Adaboost comes in place. It is used to remove many irrelevant features. Adaboost helps in determining which features are relevant and which are irrelevant

Now Adaboost is a machine learning algorithm and due to our very limited knowledge in that subject, we will defer implementing this algorithm. In a nutshell, for a given data of faces and not faces it determines which features to be used which can best describe a face.

The algorithm then makes classifiers of the respective features. To output a positive face, the image should be able to pass through all these classifiers

4. Understanding the Cascade Classifier

Now, instead of *detecting* a face, the algorithm first discards all the *non-faces*. It does so by the help of the classifiers obtained from training models.