Face Detection Using Cascade Classifier and Performance Improvement using Image Enhancement

Name: Guru Sarath Thangamani UIN: 829009551

***Abstract*—This report discusses the implementation of Viola-Jones algorithm for face detection. Along with the implementation of the above-mentioned algorithm, I have also discussed various image enhancement techniques that overcomes the short comings of this algorithm. This significantly reduces the false positive rates of Viola-Jones algorithm. For the improvement of face detection performance, thresholding and histogram equalization is used. To improve the processing speed, I have incorporated skin color detection as an add-on to the afore-mentioned algorithm.**

Keywords — Viola-Jones algorithm, Image Processing, Computer Vision, Image Enhancement, skin color detection, sliding window, Image Pyramid, histogram equalization, Haar features, cascade classifier, machine learning, Integral Image, Ada Boost.

I. INTRODUCTION

Face detection is one of the most common tasks that is required in image processing and computer vision. Face detection is usually done on streaming videos; hence we require a very fast algorithm to detect faces. Viola-Jones algorithm is used to detect faces in an image. Although the algorithm was designed to detect faces on a still image, the algorithm can be extended to face detection in videos by capturing and analyzing frames (still images) of the video. This algorithm takes long time to train, however it the prediction speed of the algorithm is very fast. This algorithm is a binary classification, machine learning algorithm.

II. VIOLA-JONES HAAR CASCADE ALGORITHM

The algorithm is a machine learning algorithm. Due to the nature of the algorithm, it can only be used for binary classification. However, multiclass classification can be achieved by training multiple models for different classes of images and then using the confidence probability to classify multiple classes.

Steps involved in the learning process:

1. Create a set of Haar features that are possible in a 24x24 window.
2. Calculate the values of the features on all the images in the dataset.
3. Initialize and normalize the weights of each weak classifier.
4. Select the best weak classifier (based on the weighted error)
5. Update the weights based on the chosen classifiers error
6. Repeat steps 2–4 T times where T is the desired number of weak classifiers
7. Finally compile a strong classifier from the set of weak classifiers. (This process is known as boosting)

Steps involved in the prediction process:

1. Creating the integral Image
2. Calculating the Haar features (Using the integral image)
3. Passing the feature through the cascade of weak classifier.
4. The image is declared as not a face even if one of the weak classifiers says that the image is not a face.

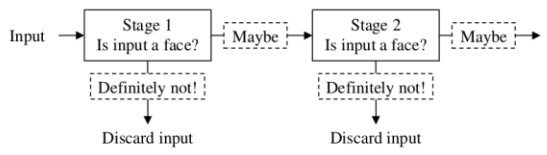


Figure 1: Shows how the cascade of weak classifiers make decision during the prediction process

III. SLIDING WINDOW

In an image, human face can be present in any part of the image; Hence, an 2D sliding window has to be implemented to scan all the M x N rectangular regions of the image.

Tunable parameters in sliding window:

1. Size of the window
2. Stride of the window

In the original algorithm proposed by Viola and Jones, the window size was fixed to 24 x 24.
Increasing stride causes the algorithm to run faster, but the chance of missing the faces in the image also

increases. Usually a stride of 5 pixels works fine in most cases.

IV. Image Pyramid

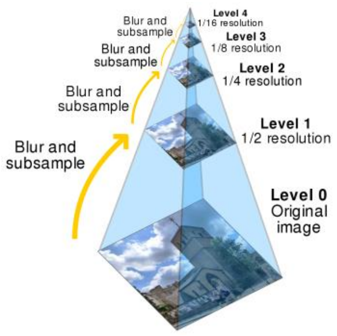


Figure 2: Image Pyramid

The sliding window procedures solves the problem of face being located in any portion of the image. However, it does not solve the problem of different sizes of faces in the image. In order to detect faces of different sizes in the image there are two approaches to solve this problem:

1. Scale the Haar features to different sizes and run the algorithm multiple times.
2. Scale the image to different sizes and run the algorithm multiple times.

Usually the image is scaled to multiple sizes, this process is known as generating the image pyramid. In the original algorithm implemented by Viola and Jones the scale factor of 1.5 was used.

V. Haar Features

Haar features are good at detecting edges and lines. Haar features looks like a strip of black and white pixels. Haar feature on an image is calculated by summing the white and black regions of the portion overlapped by the Haar feature and then subtracted. The order in which the subtraction is done is not an issue, whatever be the order, same order must be maintained throughout the process.

Value = Σ (pixels in black area) - Σ (pixels in white area)

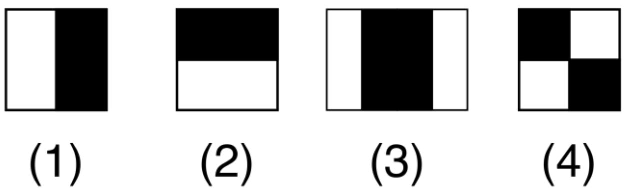


Figure 3: Haar feature (1) and (2) are edge features, Haar feature (3) and (4) are vertical and diagonal line features.

Properties common in all images of human faces:

These are Haar features used by Viola and Jones which eliminates more than **60%** of incorrect sections are:

- The eye region is darker than the upper-cheeks.
- The nose bridge region is brighter than the eyes.

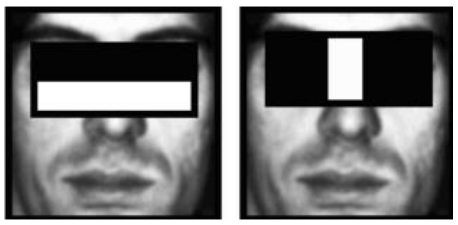
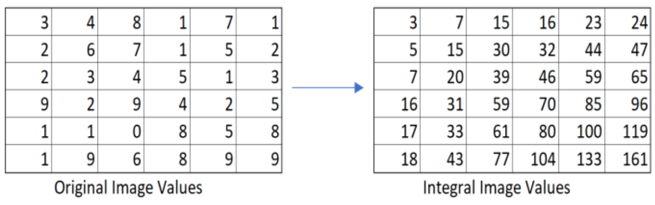


Figure 4: Eye region and Nose bridge region Haar features

VI. Integral Image

Haar features can be quickly calculated using the idea of integral image. Pixel values summation operation which is required in Haar feature calculation can be quickly performed using the integral image of the original image.

$$\text{Integral Image}(p,q) = \sum_{x=1}^{p}\sum_{y=1}^{q} Image(x,y)$$



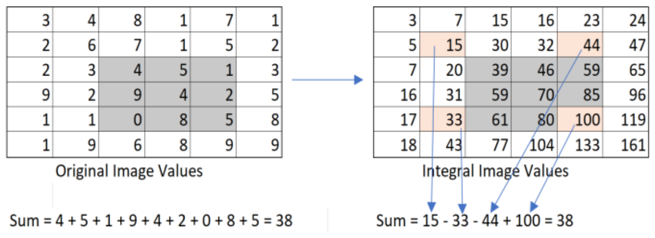
3	4	8	1	7	1
2	6	7	1	5	2
2	3	4	5	1	3
9	2	9	4	2	5
1	1	0	8	5	8
1	9	6	8	9	9

Original Image Values

3	7	15	16	23	24
5	15	30	32	44	47
7	20	39	46	59	65
16	31	59	70	85	96
17	33	61	80	100	119
18	43	77	104	133	161

Integral Image Values

Figure 5: Image and its integral image.



3	4	8	1	7	1
2	6	7	1	5	2
2	3	4	5	1	3
9	2	9	4	2	5
1	1	0	8	5	8
1	9	6	8	9	9

Original Image Values

3	7	15	16	23	24
5	15	30	32	44	47
7	20	39	46	59	65
16	31	59	70	85	96
17	33	61	80	100	119
18	43	77	104	133	161

Integral Image Values

Sum = 4 + 5 + 1 + 9 + 4 + 2 + 0 + 8 + 5 = 38

Sum = 15 - 33 - 44 + 100 = 38

Figure 6: Calculation of sum pixel values using an integral image.

A. Improving performance using Thresholding and Histogram Equalization:

The Haar cascade involves calculating the Haar features from the image. It is quite intuitive from the binary nature of Haar kernel that thresholding the image prior to providing it as input to the algorithm reduces the false positive rates of detection.

The Haar cascade algorithm also requires contrasting facial features for easy detection. Before thresholding the image, histogram equalization or contrast stretching can be performed prior to thresholding to improve the detection.

Results below show how thresholding a grayscale image can improve the performance of classifier.

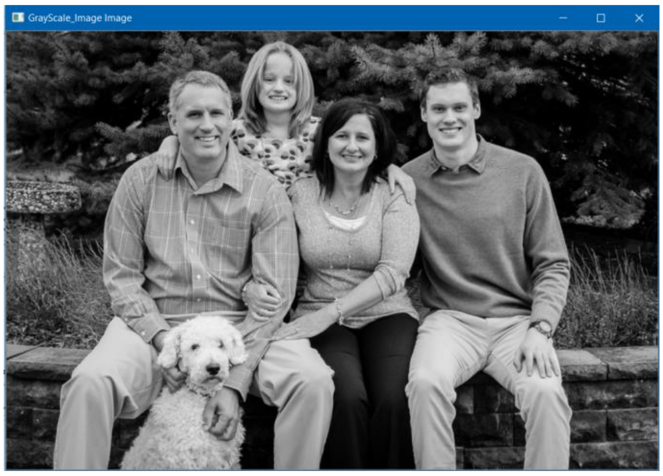


Figure 7: Input Grayscale image

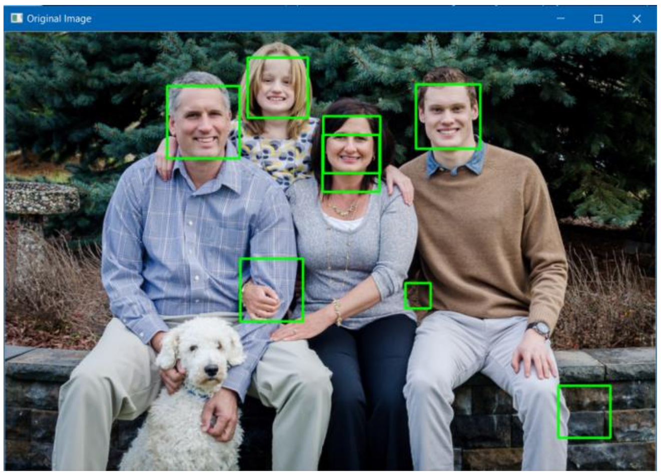


Figure 8: After running face detection on the grayscale image. You can see that there are 4 wrong detections in this single image.

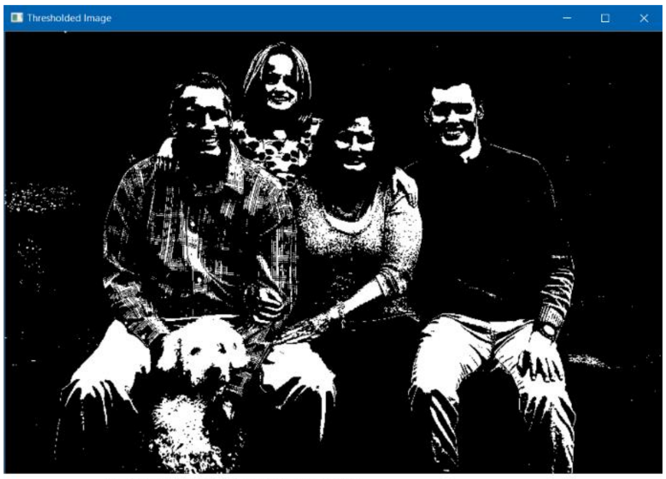


Figure 9: Input Thresholded image (Threshold value = 180)

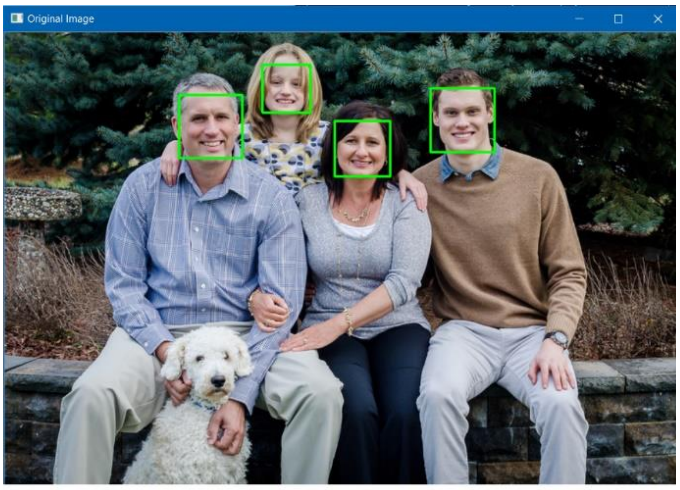


Figure 10: After running face detection on the thresholded image

B. Algorithm speed up using skin colour detection:

Color image presented in **YCrCb** color space the location of the chrominance values of the color of the human skin are narrowly and consistently distributed not only for the individual image or person, or skin color of the same human race, but over all possible varieties of the listed. The apparent difference in skin color perceived by viewers is mainly due to the darkness or fairness of the skin; these features are characterized by the difference in the brightness of the color, which is governed by Y but not Cr and Cb. [3]

Ranges R (Cr) = [133, 173] and R (Cb) = [77, 127] are very robust against different types of skin color. [3]

After detecting the regions of skin color, the original image is segmented into regions containing the skin colors. This segmented image can then be given as input to the algorithm.

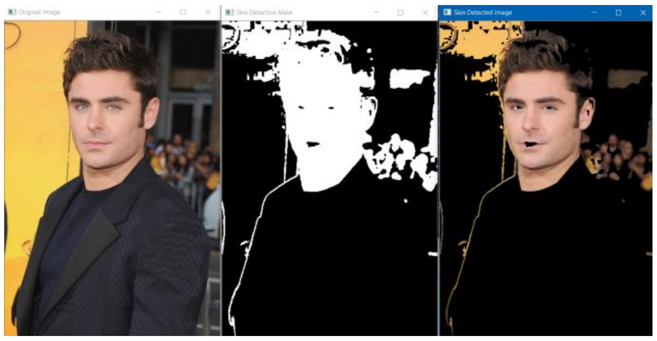


Figure 11: These three figures shows the output of the skin detection algorithm. We run the classifier only on the regions where skin is detected to speed up the detection process.

VII. Implementation Notes

The code for this project was implemented using python programming language. Additionally, OpenCV and imgutil libraries have to be installed as dependencies to run the code.

RunFullDetection.py – The code in this file runs the face detection on the input image and outputs only the final output. The input image is specified inside the file.

SlidingWindowDetector.py – The code in this file implements the sliding window algorithm and runs the cascade classifier on every sub window of the image.

ExtractSkinRegions.py – This file contains a function that takes an image as input and runs the skin color extraction algorithm on the image.

GenerateIntegralImage.py – This file contains a function that takes an image as input and outputs a numpy array, which is the integral image of the input image.

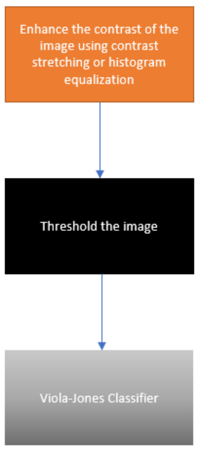

Enhance the contrast of the image using contrast stretching or histogram equalization
Threshold the image
Viola-Jones Classifier

Figure 12: This figure shows the general procedure followed to perform face detection with low false positive rates

VIII. Results

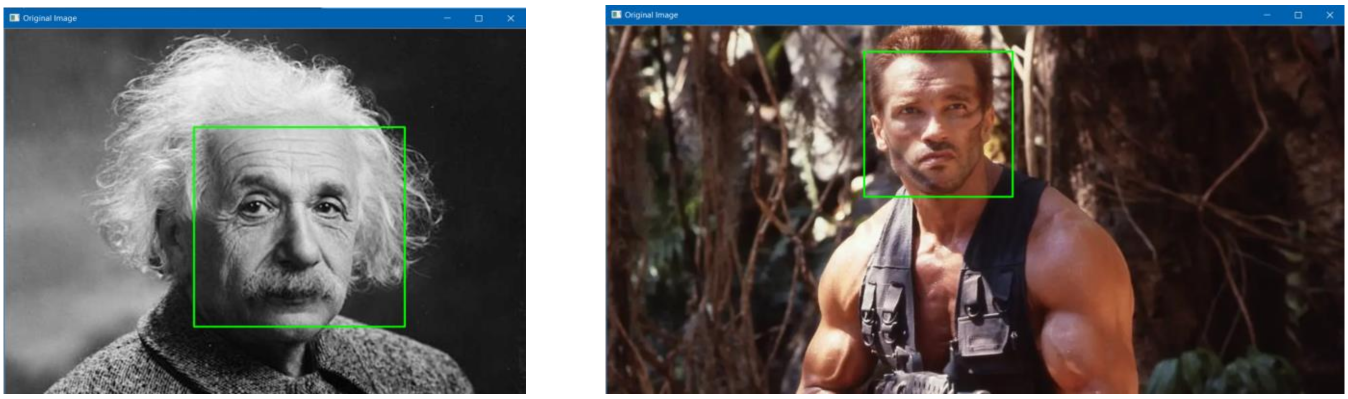


Figure 13: After running face detection on an image with only one face

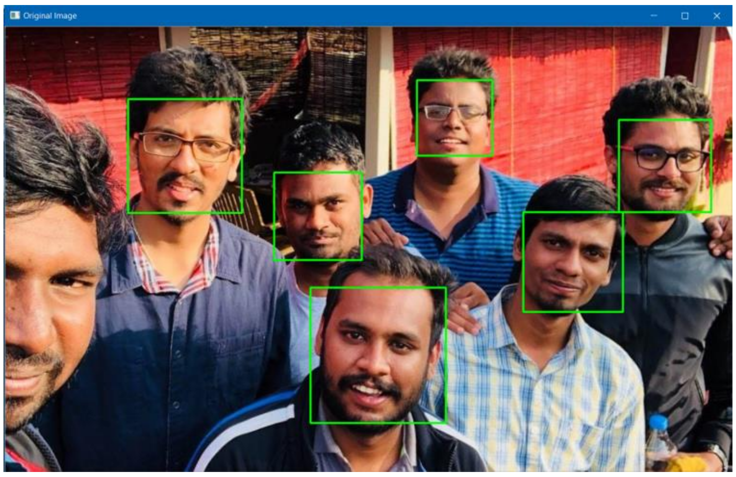


Figure 14: After running face detection on an image with many faces

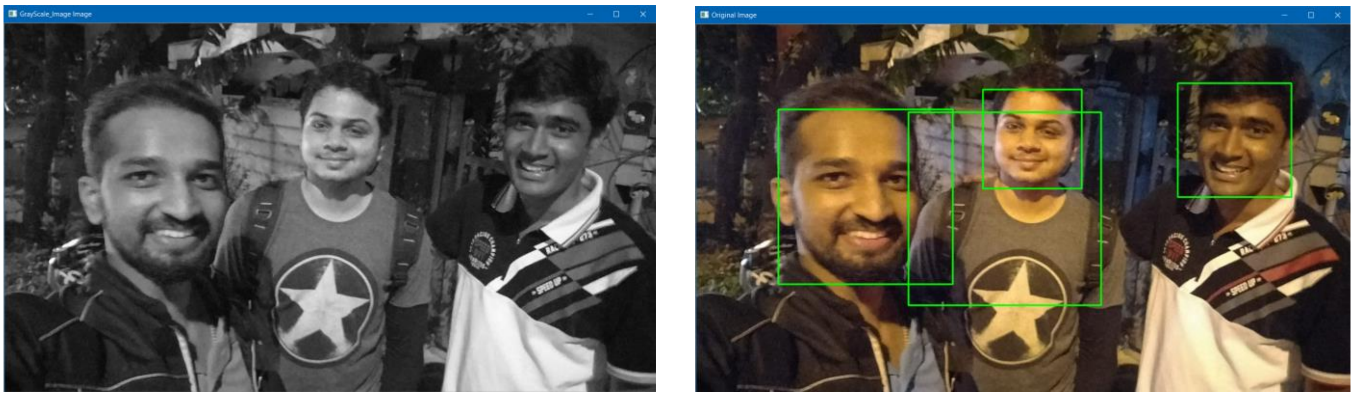


Figure 15: Left – Grayscale image; Right – After running face detection on the left image

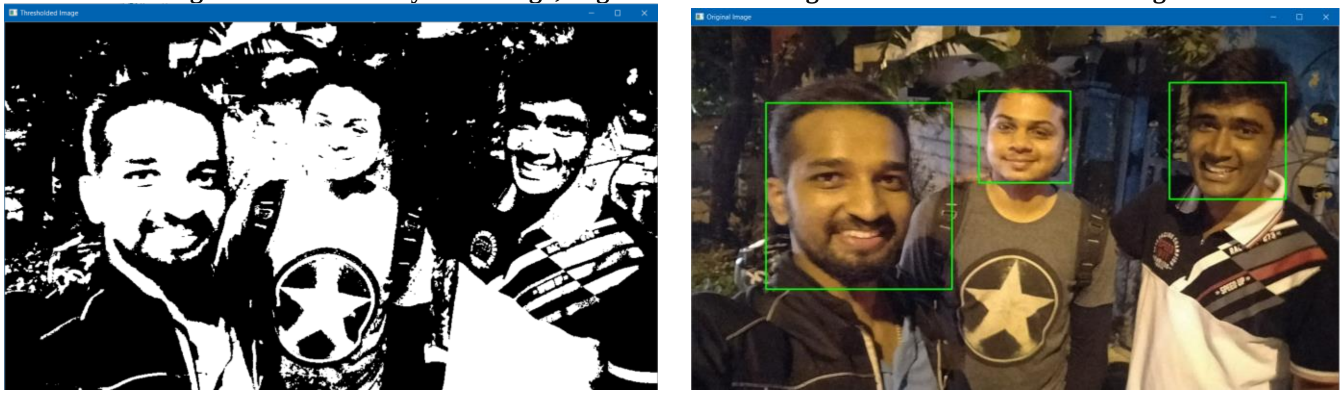


Figure 16: Left – Thresholded image (Threshold value = 80); Right – After running face detection on the left image

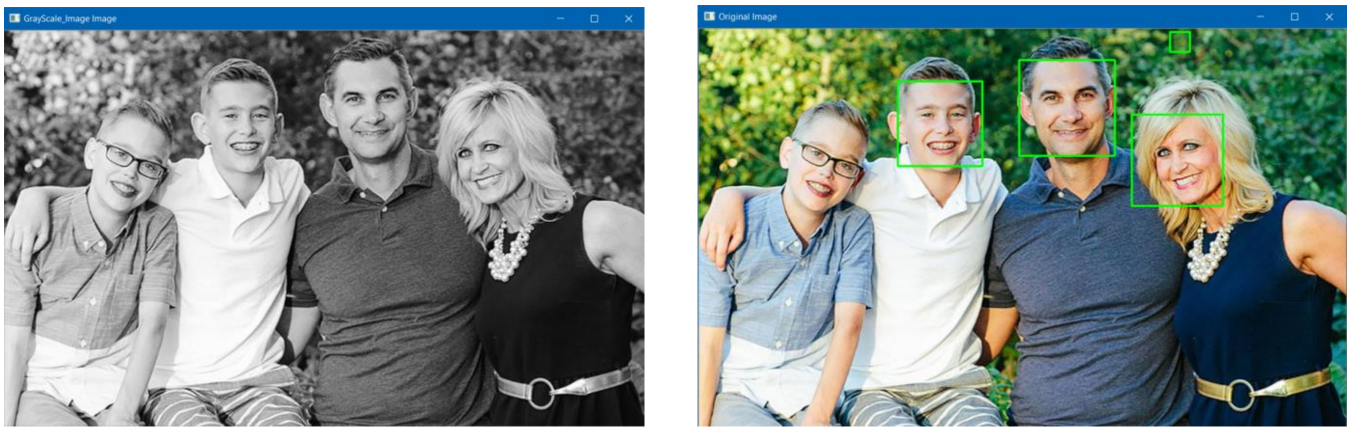


Figure 17: Left – Grayscale image; Right – After running face detection on the left image

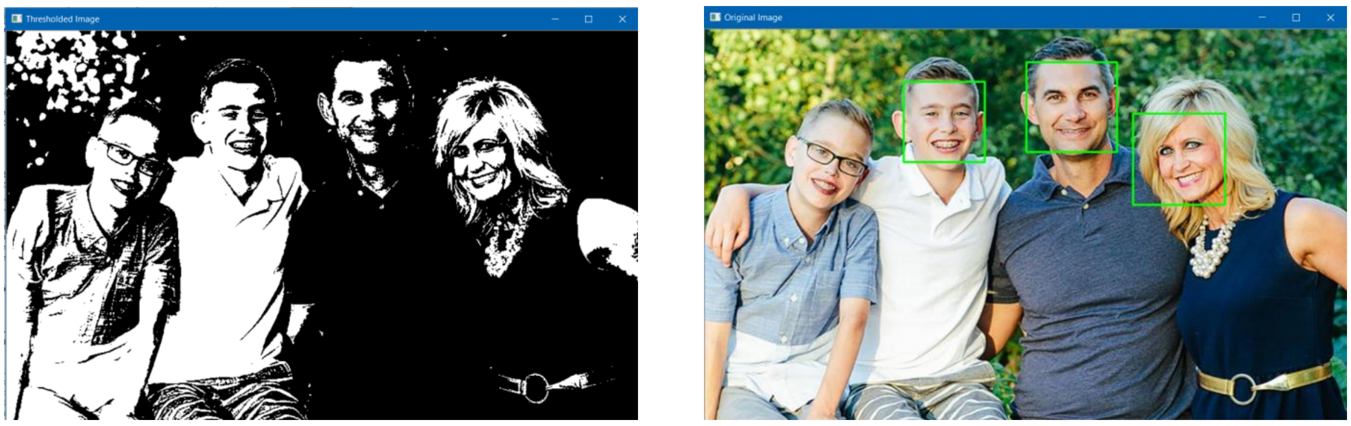


Figure 18: Left – Thresholded image (Threshold value = 180); Right – After running face detection on the left image

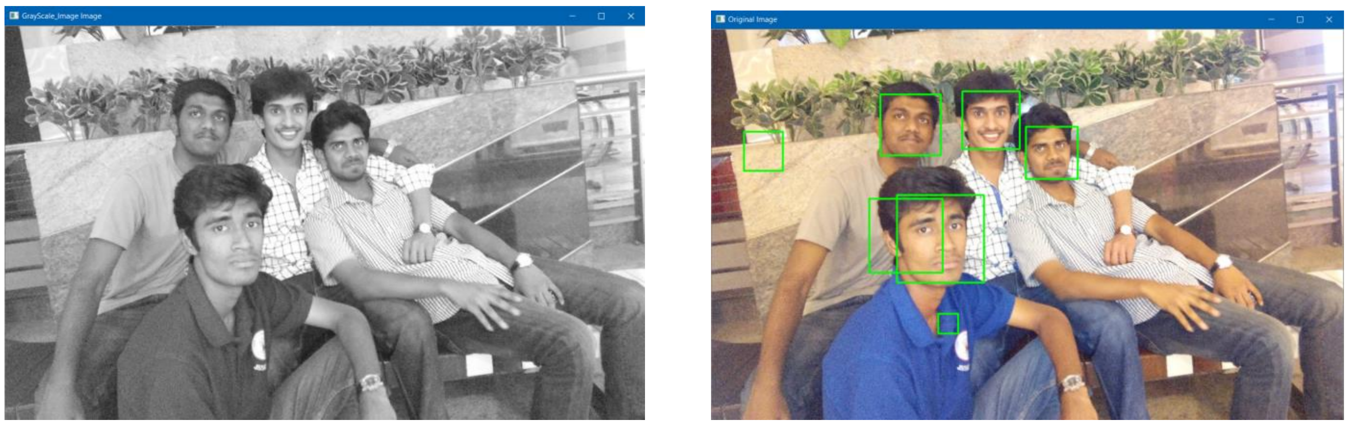


Figure 19: Left – Grayscale image; Right – After running face detection on the left image

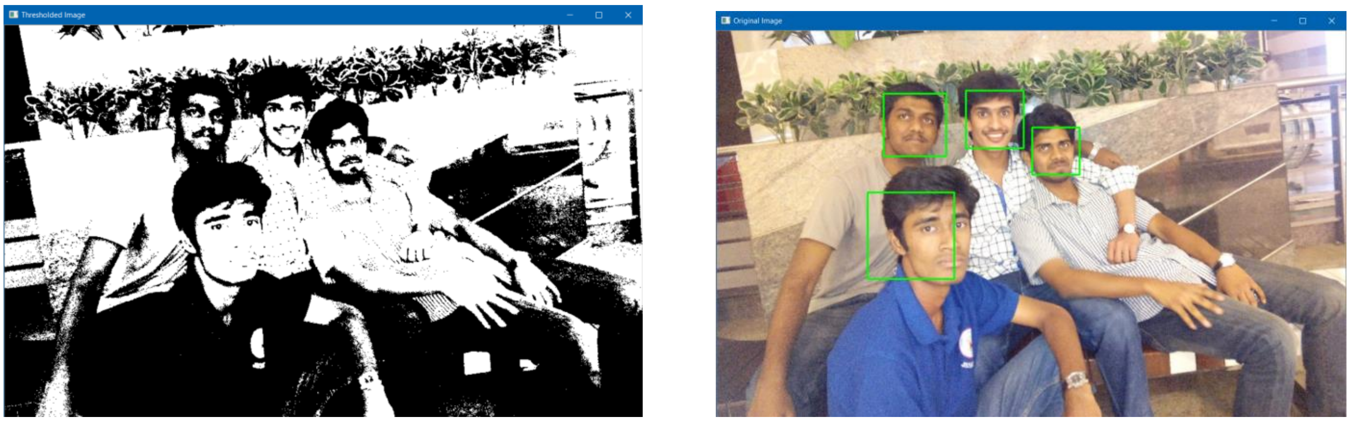


Figure 20: Left – Thresholded image (Threshold value = 150); Right – After running face detection on the left image

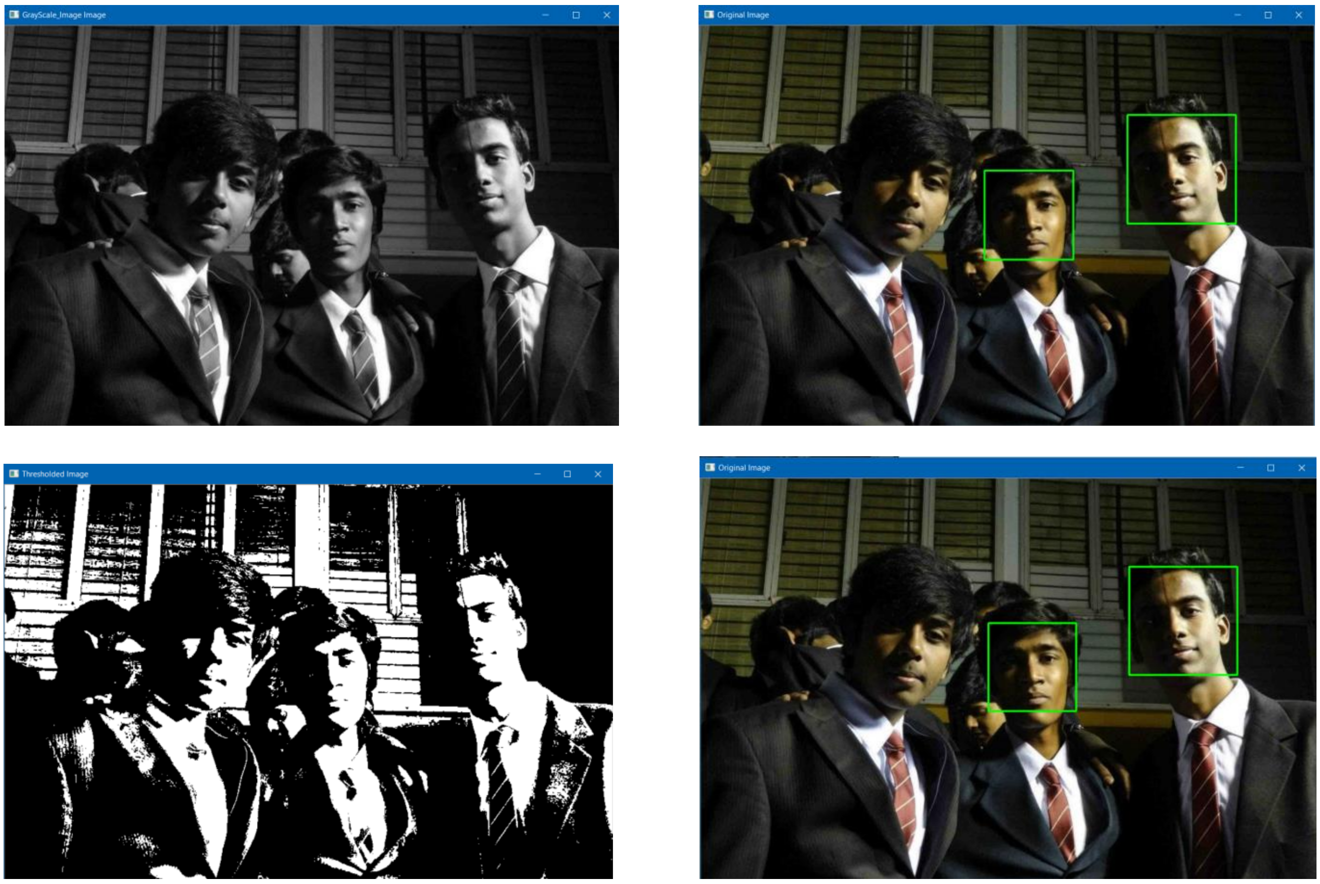
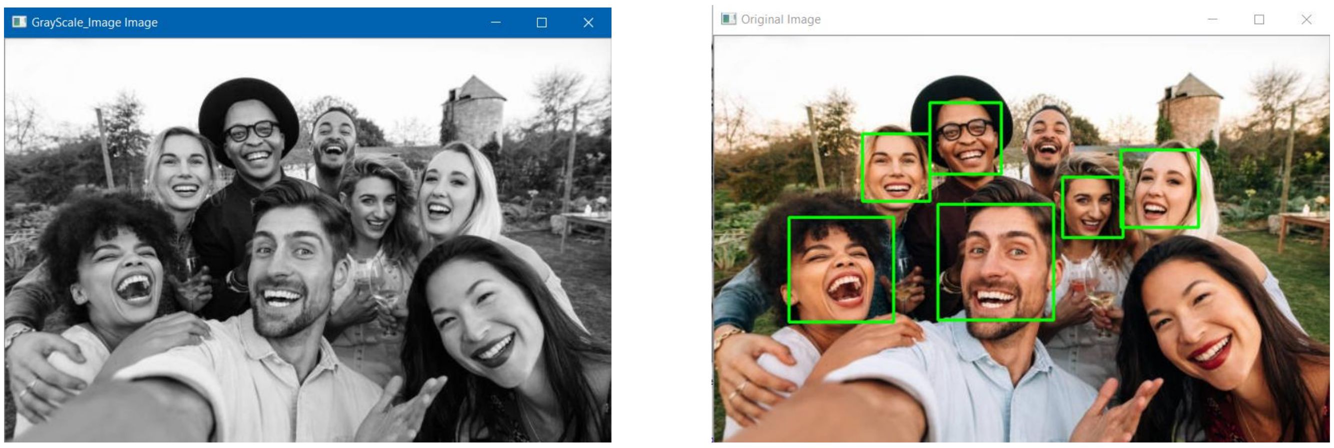


Figure 21: In the above example we can see that thresholding does not always produce better results



IX. Performance

Total Number of faces = 34

Performance before image Thresholding	
Number of Correct detections	30
Number of Wrong detections	9
Accuracy = 88.2% False Positives = 9	

Performance after image Thresholding	
Number of Correct detections	31
Number of Wrong detections	1
Accuracy = 91.1% False Positives = 1	

We can clearly see from the above tables that thresholding significantly reduces the false positive rate of the algorithm.

X. Limitations

The main limitation of the Haar cascade classifier for face detection is that it does not detect faces in all angles and a new model has to be trained again to detect faces in different views and angles. Trying to generalize the algorithm to detect faces in all angles and views significantly reduces the performance of the classifier. More advance classifier algorithms are required to detect faces from all views and angles. Training time for the cascade of the classifier is very high.

XI. Conclusion

Viola-Jones algorithm for object detection is a robust algorithm for detection of objects in an image. Due to its high speed of detection and low false positive rates it is widely used in many portable devices like cellular phones and digital cameras.

As discussed in this project report, the speed of prediction process can be further improved by analysing only the regions containing the color of skin (Works only if the input is a color image).

False positives rates can be significantly improved by thresholding the image before feeding it as input to the algorithm.

XII. References

[1] Paul Viola and Michael Jones. Rapid Object Detection using a Boosted Cascade of Simple Features.

[2] S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat, and J. Jatakia. Human Skin Detection Using RGB, HSV and YCbCr Color Models.

[3] Ludmila Natanzon. Performance improvements of Haar Cascade Classifier for face detection.

[4] https://towardsdatascience.com/whats-the-difference-between-haar-feature-classifiers-and-convolutional-neural-networks-ce6828343aeb.

[5] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

[6] https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework.