Fast Frontal Face and Eye Detection using Viola-Jones Object Detection, ICS3206, Course Project 2020

Liam Attard [0299300L]

I Viola-Jones technical discussion

A. Introduction

The Viola-Jones paper [1] describes a machine learning technique that achieves a fast and accurate object recognition method that does not base itself on deep learning. This report is still relevant today, especially for face detection and has been cited around 22,000 times. The technique used achieves itself from particular following three steps:

- The image reproduces itself using a summed-area table referred to as the integral image.
- The target object's principal features are picked out from a training set using an algorithm based on AdaBoost and generates efficient classifiers.
- The image passes into numerous filters, referred to as a ”cascade structure” which is, in essence, a degenerate decision tree.

This paper's detector was tested on 384 by 288 images at 15 frames per second and accurate, irrelevant to facial features and ethnicity. The prompt detection of this technique is what makes it ahead of other methods.

B. The Integral Image

1) Haar-Like Features

Haar-Like Features are rectangular points marked over an image, as shown in 1 where the sum of pixels in the white rectangle subtracts from the grey rectangle's sum. By performing these calculations on raw image values, the result can take a significant amount of time. By calculating the integral image, these rectangular values can be performed quickly and in constant time. Figure 2 Shows different ways of calculating the features, (a) shows two-rectangle features, (b) shows three-rectangle features and (c) shows a four-rectangle feature.

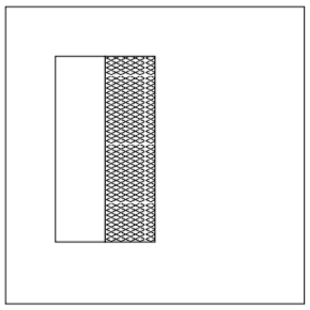


Fig. 1. Shows an example of a Haar features

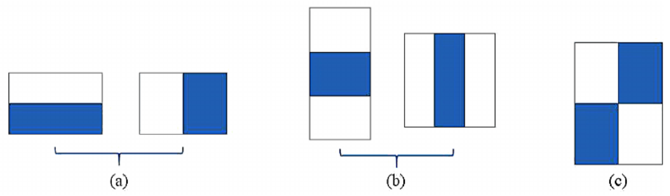


Fig. 2. Different Ways of calculating Haar Features

2) Calculating The Integral Image

The integral image is found by iterating over each pixel and computing its new value. This is obtained by calculating the sum of pixels above and to its left. The original pixel i(x,y)'s integral image ii(x,y) can be found using the following equation:

$$ii(x, y) = \Sigma_{(x \leq x, y' \leq y)}$$

Example of original image and its integral image.

1	5
2	4

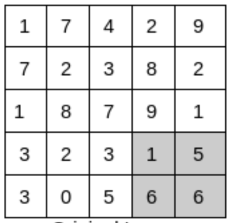
Original Image

1	6
3	12

Integral Image

3) Calculating the Sum of Any Pixel Value From the Integral Image

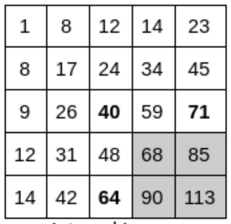
Consider the following matrix:



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

Original Image

The sum of the greyed out area is 1+5+6+6 which is **18**. Now consider its integral image:



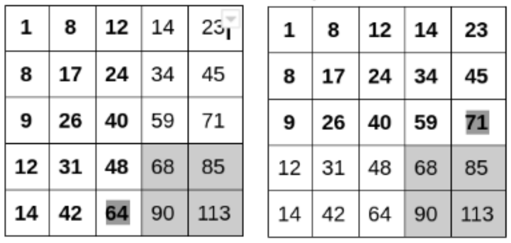
1	8	12	14	23
8	17	24	34	45
9	26	**40**	59	**71**
12	31	48	68	85
14	42	**64**	90	113

Integral Image

To Calculate the sum of the greyed out area subtract the summation of the unwanted areas's corner values, in this case:

113 - 64 - 71 which is -22 as shown in figure 3.

Fig. 3. Integral Image Corners



1	8	12	14	23
8	17	24	34	45
9	26	40	59	71
12	31	48	68	85
14	42	64	90	113

1	8	12	14	23
8	17	24	34	45
9	26	40	59	71
12	31	48	68	85
14	42	64	90	113

Fig. 3.

Then re-add the corner values of the areas that have been taken off twice in this case

+40 which results to **18**.

Although in this case, it did not make sense to calculate the integral image given how small the size of the original image is, in large pictures, this would save much time.

C. Learning Classification Functions Using AdaBoost

Over 180,000 features can be calculated in an image given a small detector of size 24x24 pixels. This number of features results in both slow classification and overfitting. To avoid such problems a boosting algorithm (AdaBoost) [2] is used to select a small set of critical Haar-like features using weak classifiers. For each of the features, the algorithm finds the function which separates them positively (contain a face) and negatively (do not contain a face) in the most optimal way ie. leaving the least possible number of values misclass Given a 24 by 24 image x with feature f_j, a weak classifier (h_j), is defined by

$$h_j(x) = 1 \text{ if } p_j f_j < p_j \theta_j$$
$$h_j(x) = 0 \text{ otherwise}$$

where θ_j is the treshold and p_j indicates the direction of the inequality sign.

1) *Resultant Features*

The figure shows the few numbers of features selected by AdaBoost. The two chief characteristics given priority are the darker eye area over the cheeks and the nose's bridge.

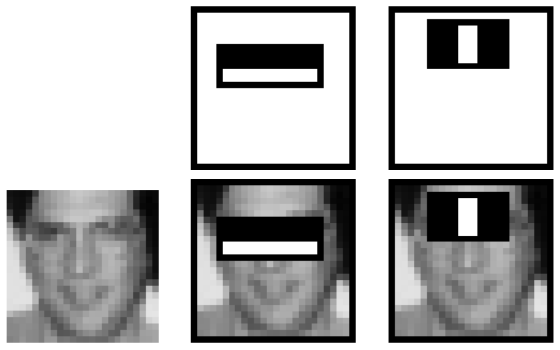
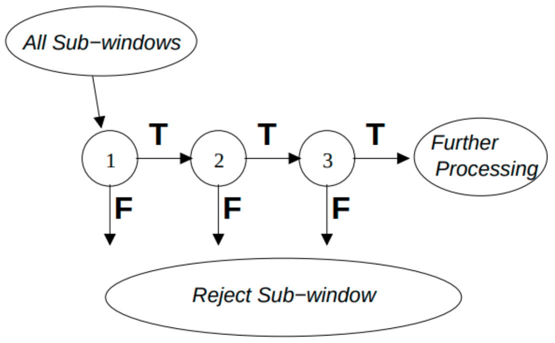


Fig. 4. Some Resultant Features

D. *Cascade of Classifiers*

The third part of the algorithm places the previous features in the form of ranking stages where an image would have to pass through all of the stages which contain the object/face. By doing so, this increases performance by avoiding to count all the negative features. If the image fails to pass through the first feature, it fails.



E. *Useful Applications for the Viola Jones algorithm*

Although the modern standard approach for detecting objects is deep learning, the viola jones algorithm is still widely used today. The technique does not require much storage and can run on low powered devices and smartphones efficiently. That is why two modern real-life examples include several digital cameras, and the Snapchat face filters. [3]

II Methodology

A. *Artefact 1*

1) *The Datasets*

The two datasets used to train a cascade classifier for face recognition were:

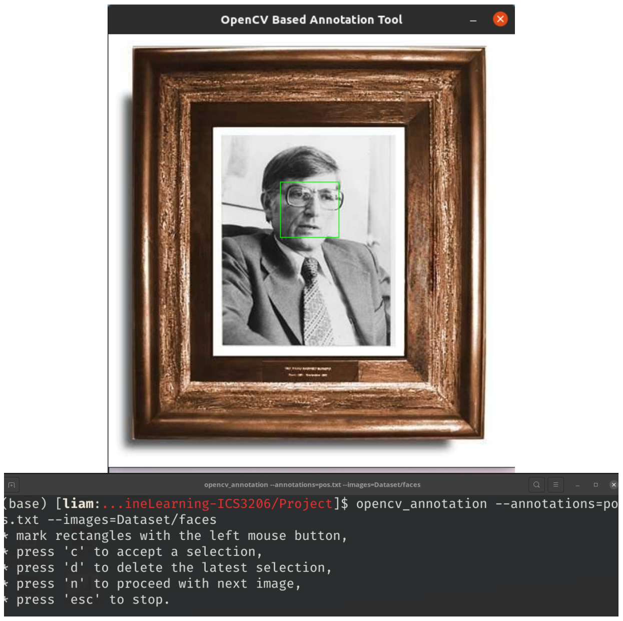
- Caltech 10,000 Web Faces [4], representing the positive values and
- The Stanford Background Dataset [5], representing the negative values.

The first dataset produced by Caltech University consists of images of people gathered from the web by searching for common given names into Google images. The second dataset consists of 715 outdoor images chosen from public datasets.

B. *Training Data preparation*

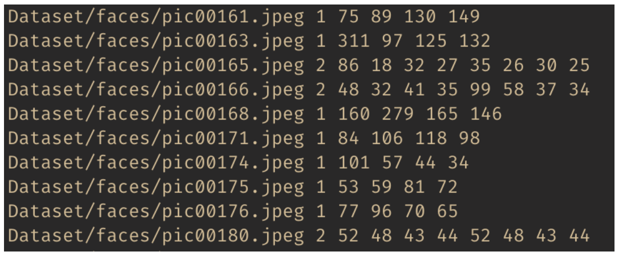
1) *Face Annotation*

The training data preparation requires item two text files containing the directory of each positive and negative images and the coordinates of the positive images' faces. OpenCV's annotation tool was used to help with annotating around 700 faces as can be seen in figure 5. Few lines from the positive text file can be seen in figure 6.


OpenCV Based Annotation Tool
opencv_annotation --annotations=pos.txt --images=Dataset/faces

```
(base) [liam:...ineLearning-ICS3206/Project]$ opencv_annotation --annotations=po
s.txt --images=Dataset/faces
* mark rectangles with the left mouse button,
* press 'c' to accept a selection,
* press 'd' to delete the latest selection,
* press 'n' to proceed with next image,
* press 'esc' to stop.
```

Fig. 5. OpenCv Annotation Tool in progress



```
Dataset/faces/pic00161.jpeg 1 75 89 130 149
Dataset/faces/pic00163.jpeg 1 311 97 125 132
Dataset/faces/pic00165.jpeg 2 86 18 32 27 35 26 30 25
Dataset/faces/pic00166.jpeg 2 48 32 41 35 99 58 37 34
Dataset/faces/pic00168.jpeg 1 160 279 165 146
Dataset/faces/pic00171.jpeg 1 84 106 118 98
Dataset/faces/pic00174.jpeg 1 101 57 44 34
Dataset/faces/pic00175.jpeg 1 53 59 81 72
Dataset/faces/pic00176.jpeg 1 77 96 70 65
Dataset/faces/pic00180.jpeg 2 52 48 43 44 52 48 43 44
```

Fig. 6. Examples of the Positive Value File

2) *Positive Vector File*

A vector file generated from the positive values' text file is also required for cascade training which was generated using OpenCV's createSamples program.

C. Cascade Training

The next step is the boosted cascade of weak classifiers based on the positive and negative dataset prepared beforehand using OpenCV's train cascade function. Several models have been generated by changing the parameters' values to meet a good result. The evaluation section will show the model's results. The chosen model for this project which was generated by testing out different parameters contains the following parameters:

```
$opencv_traincascade -data cascade
-vec pos.vec -bg neg.txt -w 24 -h 24
-numPos 800 -numNeg 425 -numStages 15
-precalcValBufSize 3000 -precalcIdxBufSize
3000 -maxFalseAlarmRate 0.4
```

- **Number of stages: 15 (Started with 10 and increased to 15 for more accuracy but also keeping in mind overfitting)**
- **Maximum False Alarm Rate: 0.4**
- **Minimum hit Rate: 0.995**
- Number of positive samples: 800 (The Number of manualy annotated samples)
- Number of negative samples: 425
- Sample width: 24
- Sample weight: 24
- Number of unique features given window Size [24,24] : 162336
- Acceptance ratio break value: -1
- Stage type: BOOST
- Feature Type: HAAR
- Boost Type: GAB
- Weight Trim Rate: 0.95
- Max Depth: 1
- Max Weak Count: 100
- Mode: BASIC

D. Artefact 2

Artefact 2 The two downloaded models haarcascade_frontalface_default.xml, haarcascade_eye.xml for the frontal face and eyes, respectively, are tested using OpenCV's **cascadeClassifier** function. Check for the results and comparisons in the evaluation section.

III Evaluation and Results

A. Artefact 1

The model's final stage contained **33 weak trees** with a **hit rate of 0.99625**, a **false alarm rate of 0.383529** and a **negative acceptance ratio 425 : 0.000470985**. In general, the model produces very good results as seen in figures 10, 11 but produces some false faces when there are complicated backgrounds as seen in figure 9.

1) Some Good Results

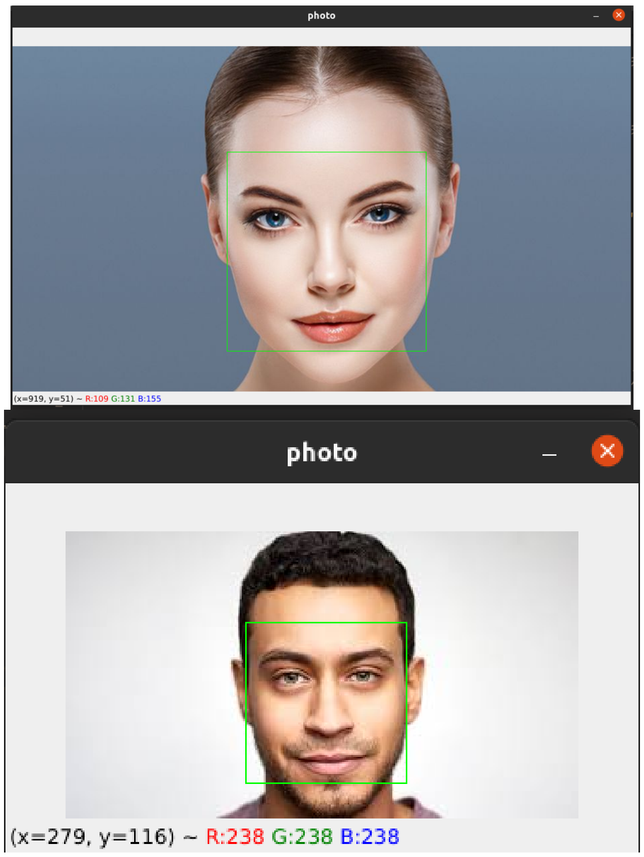


photo
(x=919, y=51) ~ R:109 G:131 B:155
photo
(x=279, y=116) ~ R:238 G:238 B:238

Fig. 7. Testing Results, (Images were not included in the dataset)

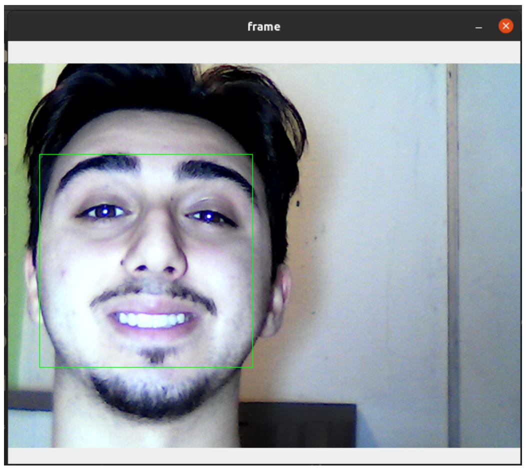


Fig. 8. Webcam Result

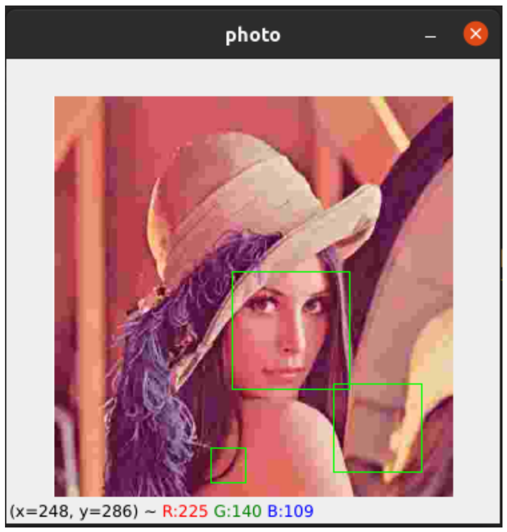


Fig. 9. Result with some Errors

2) *Classification Metrics*

some metrics when given 4 photos with a total of 24 faces.

Classification Accuracy

$$Accuracy = \frac{No\ of\ Good\ Predictions}{Total\ No\ of\ Predictions} \tag{1}$$

27/20=0.64

Prediction outcome

		p	n	total
	p′	26	1	P′
actual value	n′	13	All Background 24x24 segments	N′
	total	P	N	

B. *Artefact 2 results*

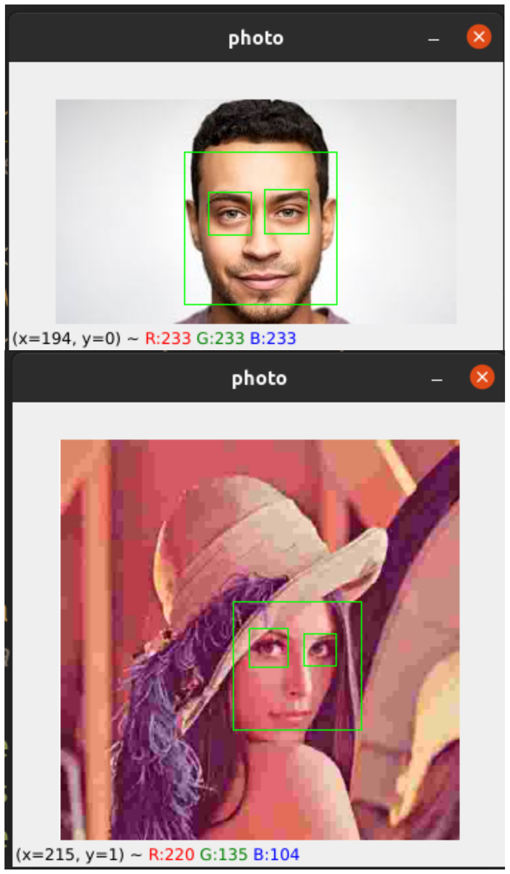


photo
(x=194, y=0) ~ R:233 G:233 B:233
photo
(x=215, y=1) ~ R:220 G:135 B:104

Fig. 10. Testing Results, (Images were not included in the dataset)

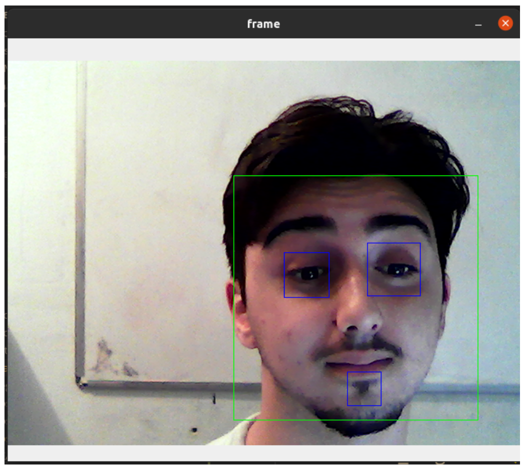


Fig. 11. Webcam Result

References

[1] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I.

[2] Y. Freund, R. Schapire, and N. Abe, “A short introduction to boosting,” *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[3] M. Bansal and R. Sharma, “Journal of Analysis and Computation (JAC) FACE DETECTION USING VIOLA-JONES ALGORITHM & CONVOLUTIONAL NEURAL NETWORKS,” Tech. Rep. [Online]. Available: www.ijaconline.com,

[4] M. Fink, R. Fergus, and A. Angelova, “Caltech 10,000 web faces,” *URL http://www. vision. caltech. edu/Image_Datasets/Caltech_10K_WebFaces*, 2007.

[5] “Stanford Background Dataset — Kaggle.” [Online]. Available: https://www.kaggle.com/balraj98/stanford-background-dataset