|  |  |
| --- | --- |
|  | Documentation |
|  |  |
|  | Liam Bugeja Douglas  ICS2207 Course Project  12/27/21 |

Contents

[State of Completion 2](#_Toc91784454)

[Content of Folder 3](#_Toc91784455)

[Technical Discussion on Viola-Jones Method 4](#_Toc91784456)

[Real World Application of Viola-Jones Algorithm 10](#_Toc91784457)

[Methodology 11](#_Toc91784458)

[Evaluation 12](#_Toc91784459)

[Comparing Viola Jones to YOLOv5 14](#_Toc91784460)

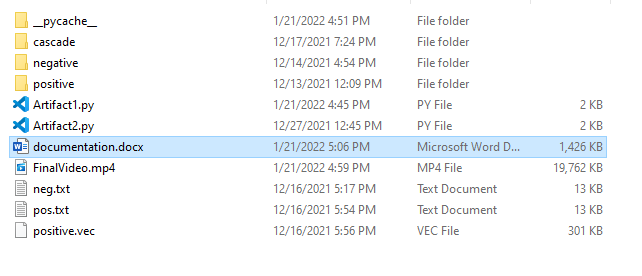
[Sources 16](#_Toc91784461)

Plagiarism Form……………………………………………………………17

## State of Completion

|  |  |
| --- | --- |
| Item | Completion |
|  |  |
| Artefact 1 | Completed |
| Artefact 2 | Completed |
| Artefact 2 – real time face/eye tracking | Completed |
| Technical Discussion on Viola-Jones method | Completed |
| Experiments and Evaluation | Completed |

## Content of Folder



* cascade: Contains the model of the cascade classifier used in Artifact 1 as well as the training data.
* negative: Contains the negative images which are used to train the model.
* positive: Contains the positive images which are used to train the model.
* Artifact1.py: The model which is trained to detect faces using our own data.
* Artificat2.py: The model which is used to detect faces and eyes in real time.
* documentation.docx: The documentation of the project.
* FinalVideo.mp4: Real time face and eye tracking of Artifact 2.
* neg.txt: Contains the path of all negative images, this is used when training Artifact 1.
* pos.txt: Contains the path of all positive images and the bounding rectangles in the images, this is used when training Artifact 1.
* positive.vec: Contains a dataset of the positive images in a binary format, this is used in Artifact 1.

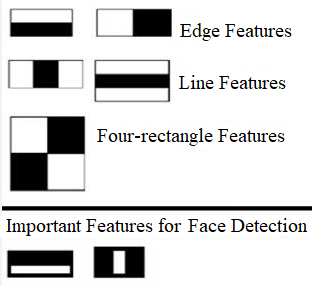
## Technical Discussion on Viola-Jones Method

Developed in the early 21st Century by Paul Viola and Michael Jones, hence the name Viola-Jones, is an object recognition framework mainly used for face detection. However, it can be trained to detect other objects.

The Viola-Jones algorithm has two main stages: The first stage is to train the algorithm to identify features and patterns, the second stage it detects these features in images or real time video capture. This is done by outlining boxes in the image or video and searches for the features and patterns. These features are called Haar-like features. Each stage can be split in two different stages, Detection: 1.Haar Feature Selection and 2.Creating an Integral Image; and Training: 1.Training Classifiers and 2.Adaboost Training and Cascading.

Haar Feature Selection

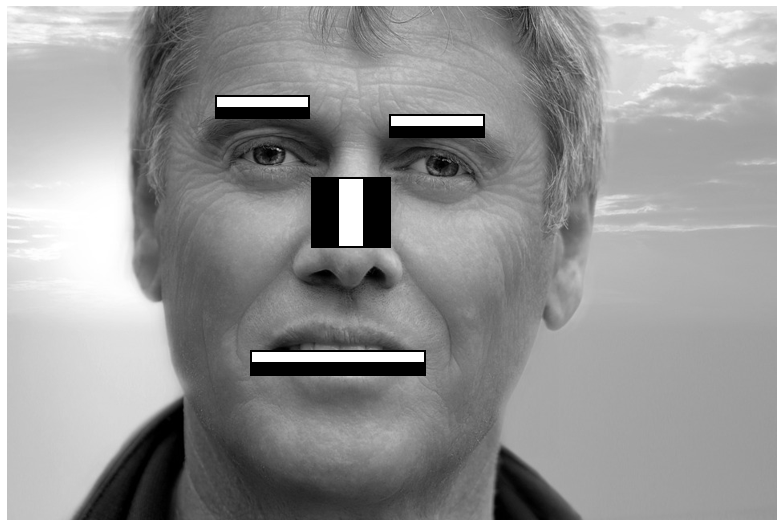
Let’s focus on Haar Feature Selection. As said previously, the machine searches for Haar-like features but what are these features? The image below shows multiple boxes with each box containing a light and dark side, which can represent facial features. Since almost all human faces share the same physique, the machine can be trained to detect them, for example the eye region is darker than the surrounding area in pixels.



The Viola-Jones algorithm uses 2 types of Haar-like features:

1. Horizontal Features
2. Vertical Features

Using these features the machine can understand what the image is and find the characteristics of the human face. The image below represents what the machine does.



Next the machine calculates a value for each feature which is almost always unique and simple to do. It converts each Haar feature into a specified grid, and each square represents a pixel with a value between 0 and 1, the darker the pixel the higher the value. Finally, it adds up the total value of the black and white pixels separately and subtracts them.

Let’s take the mouth of the above image as an example, I will create a simple grid and do the calculation, the values will be randomly assigned.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.2 | 0.3 | 0.3 | 0.2 | 0.1 | 0.1 | 0.3 | 0.3 | 0.2 |
| 0.2 | 0.3 | 0.1 | 0.1 | 0.2 | 0.2 | 0.3 | 0.1 | 0.1 |
| 0.5 | 0.6 | 0.5 | 0.5 | 0.7 | 0.7 | 0.6 | 0.8 | 0.8 |
| 0.6 | 0.5 | 0.5 | 0.5 | 0.8 | 0.8 | 0.5 | 0.8 | 0.6 |

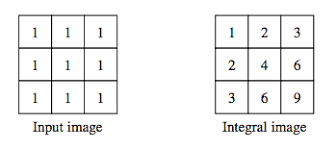
Next, we calculate the value: Black – White =

(0.5 + 0.6 + 0.5 + 0.5 + 0.7 + 0.7 + 0.6 + 0.8 + 0.8 + 0.6 + 0.5 + 0.5 + 0.5 + 0.8 + 0.8 + 0.5 + 0.8 + 0.6) - (0.2 + 0.3 + 0.3 + 0.2 + 0.1 + 0.1 + 0.3 + 0.3 + 0. 2 + 0.2 + 0.3 + 0.1 + 0.1 + 0.2 + 0.2 + 0.3 + 0.1 + 0.1) = 11.3 - 3.6

***=7.7***

Integral Image

In Haar feature selection we calculated one feature, however, the machine does this to all features present in the image as well as having a larger grid to represent all the pixels. With this in mind we can conclude that this process is time and resource consuming. So, the machine uses another grid called an integral image which speeds up this process. To calculate the integral image the machine simply takes the sum off all boxes to its left, the example below can clarify how it is done.



Next let’s say that the input image is a feature to calculate we simply add the top left and bottom right value and subtract the top right and bottom left value.

***1 - 3 - 3 + 9 = 4***

Since the Haar features are rectangular, by having the integral image values we can find the difference of the black and white pixels much easier since the total value is already calculated. This method works especially with larger images since it will save a lot of time in calculations.

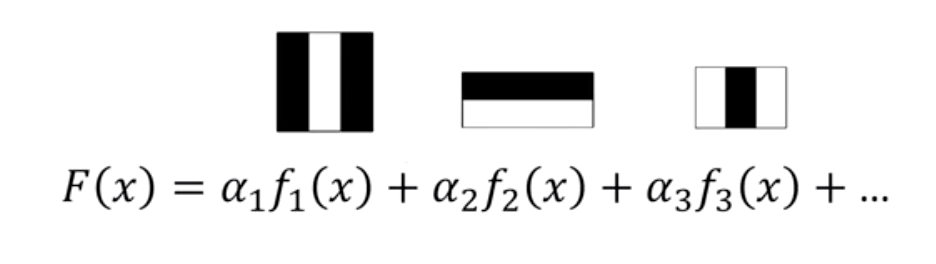
Training Classifiers

Now that detection is done, we can cover training, so when we mention training, we are actually training the machine to identify features by feeding it data in images. With this data, the machine learns from the information to predict if an image is in a certain category. Let’s take faces, the machine is fed faces and non-faces, it then finds a pattern and when it is given a random image which is not from the training set, it must identify if must be classified in the faces or non faces group. The algorithm requires a lot of data, in this project around 250 positive and 250 negative images were used.

The algorithm then works by shrinking an image into a 24x24 size and searches for the trained features in the image.

Ada Boost Training and Cascading

Training can be a lengthy process since the algorithm must be trained for all the different possibilities, thus for it to be highly accurate the model must look at all positions and combinations of the features. In the Viola Jones algorithm, the below equation is used:

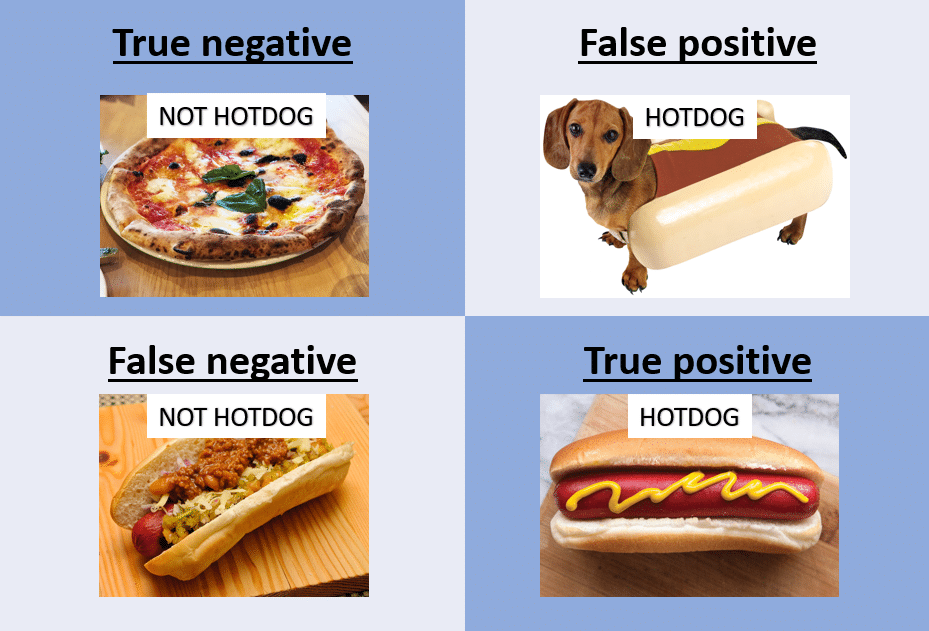


Where:

* *fi*is the feature
* *ai* is the weight of the feature
* *F(x)* Total weight of all the features

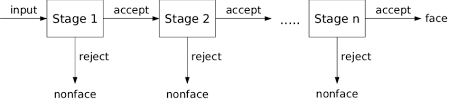
The left-hand side of the equation can be called a strong classifier whilst each feature of the right-hand side is a weak classifier. Therefore, the more features we have in our dataset the more accurate the algorithm is. However, we need to make sure that the important features are at the front in the right-hand side of the equation. This can be done by using AdaBoost training.

AdaBoost training works by giving higher weights to images which it labelled as false negatives and finds the best feature in the images to fit them in the true positive section.



When the algorithm places all the features correctly (meaning it is optimized) and is able to correctly label positive and negative images it then moves to cascading.

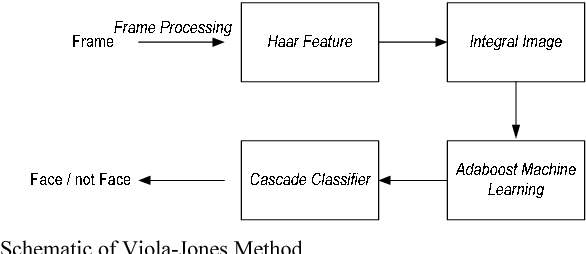
Cascading is a method in which it enhances the speed and accuracy of the model. It works by taking subwindows and compares the images by finding if it has the best feature, if not than it discards the subwindow. If it is present in the image then it checks the next feature. It continues to this for a number of features. This method speeds up the process since it discards subwindows that do not match certain features. Therefore, it can deliver results faster.



Conclusion

In conclusion, the algorithm has two main stages: training and detection. Each stage has 2 steps:

* Training: Training Classifier and AdaBoost and Cascading.
* Detection: Detecting Haar-like features and Creating Integral Image.



## Real World Application of Viola-Jones Algorithm

Despite being an outdated framework since it was released in 2001, the algorithm is still being used since it is quite powerful in real time facial-detection. Even though the algorithm takes a long time to train, its real time face detection is quite fast. However, it suffers in quite a bit of areas. Firstly, the algorithm has difficulties detecting faces at different angles and at bad lighting environments. This occurs because Haar-features do not map very well at different positions.

Currently, Snapchat’s framework uses the algorithm when applying filters on peoples’ faces. It works perfectly with Snapchat since most of the time the users place the camera close to their face, so detection is quite easy.

## 

## Methodology

Artefact 1:

First, I searched the web to find positive and negative images, for the positive images I found a dataset from the University of Massachusetts Amherst. However, for the negative images I was not so lucky so I created one from finding random images from the web.

Next, I created two folders called ‘positive’ and ‘negative’ which contained the images. After that, I made a function which given the negative image, it creates a text file which has the locations of all the negative images. I did this as well for the positive images, however, I used an opencv function called opencv\_annotation.exe which asks the user to label all the images.

Later, I created a vector file which needs to be used when training the cascade classifier. To do this I used an opencv function called opencv\_createsamples.exe

Afterwards, I trained the model by using the opencv function called opncv\_traincascade.exe.

Finally to see the model working I wrote a function called get\_trained\_model() which gets the cascade classifier and draws a blue rectangle when it detects a face. A loop is used so the process is done constantly, when the user wants to end the program, they must press ‘e’ on their keyboard.

Artefact 2:

First, I instantiated the face detection and eye detection from opencv, then to see the models working, I used my webcam and when the models detect my face or eyes it draws a green and blue rectangle respectively. A loop is used so the process is done constantly, when the user wants to end the program, they must press ‘e’ on their keyboard.

## Evaluation

Artefact 1:

I believe that the cascade classifier that I built has been trained for a perfect number of stages and has a large enough dataset that it can predict a face with high accuracy. To test the model, I used my webcam during daytime and positioned my head in different angles as well as wearing objects on my face to test the limits of the model.

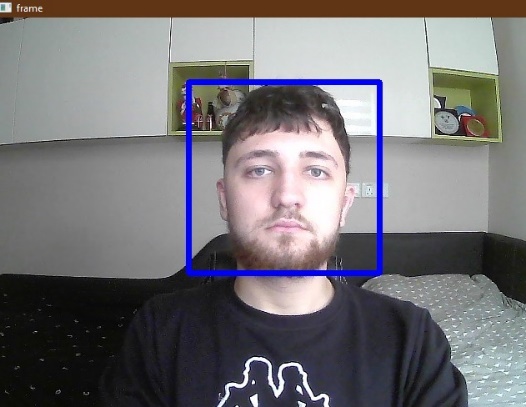
***fig 1.***

Figure 1 shows my face in perfect lighting and looking straight towards the camera. Obviously, the model easily detected my face since it had no distractions and had the perfect scenario.

***fig 2.***

Figure 2 shows my face wearing a beanie, mask and a pair of glasses. To my surprise the model was still able to detect my face even though most of my face was obstructed.

 ***fig 3.***

Figure 3 shows my face looking away from the camera, however, my facial features can still be seen. The model did not detect right away and took a second to detect my face.

*** fig 4.***

Fig 4 shows my face looking away from the camera, only my right face features can be seen. The model did not detect my face since the Viola Jones algorithm is sufficient in detecting faces at different positions.

In conclusion, the model worked as intended and the outcomes I desired were almost fulfilled. I expected that the model could detect my face even at different angles such as figure 4, however after doing some research I found out that the algorithm has a hard time detecting faces at such angles. Also, the face detection would sometimes stop for a few milliseconds and resume as if nothing happened.

Artefact 2:

The prebuilt cascade classifier was a bit more accurate when it came to facial detection. However, I did notice that it struggled when it came to eye detection. If I was not looking straight into the camera and remained still, the eye detection would bug out. Also, when wearing glasses, the eye detection would also bug out. To test this model, I used my webcam during daytime and positioned my head at different angles as well as wearing some objects to test the limit.

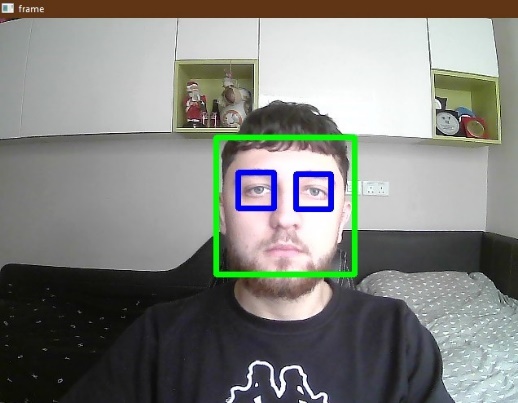
 ***fig 1.***

Fig 1 shows my face looking towards the camera, the model easily detected my face and eyes and it had no trouble tracking the features.

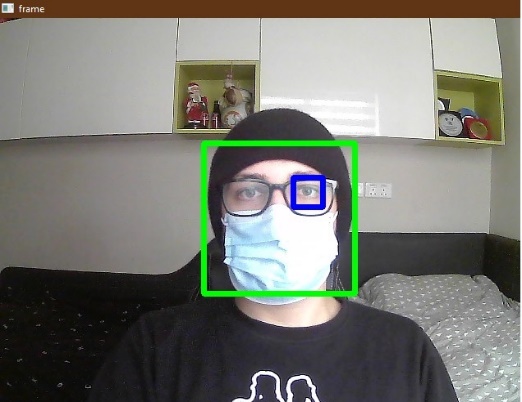
***fig 2.***

Fig 2 shows my face wearing a beanie, mask and glasses. The model had no problem detecting my face however, it had issues when it came to detecting my eyes. I tested a bit around and found out the glasses caused this issue due to lighting.

*** fig 3.***

Fig 3 shows my face not looking towards the camera. The model struggled much more than Artefact 1 when it came to detecting my face in different positions.

In conclusion, the model worked however I did expect more from it. The models eye detection suffers a bit and bugs out most of the time, especially when moving my face or wearing glasses. Face detection was much better than the Artifact 1, it would not bug out at all. In the folder a video called “FinalVideo.mp4” showcases the real-time face and eye tracking of this pre-build model.

## Comparing Viola Jones to YOLOv5

YOLOv5 is an object detection algorithm that was released in 2020 by Glenn Jocher who introduced the algorithm using Pytorch framework. YOLO is an acronym for ‘You only look once, the first version (YOLOv1) was released in 2015 as a research paper by Joseph Redmon.

The algorithm works by using real time object detection with convolutional neural networks (CNN’s), which gives faster and more accurate results than the Viola Jones algorithm. The model was also created to detect multiple objects from planes to sushi unlike Viola Jones which was created to detect facial features.



The above image displays the architecture of the algorithm, which when compared to the Viola-Jones structure is much more complex and complicated. Moreover, it is split up into three different stages:

* Backbone: A CNN that aggregates and forms image features at different granularities
* Neck: Multiple layers to put together the image features found in the backbone and pass the for prediction
* Head: Retrieves features from the neck and inputs boxes and object prediction.



The above image displays the result of using the YOLOv5 algorithm which we can see multiple objects being detected with correct classes. It shows how powerful this algorithm is when compared to the Viola Jones algorithm.

## Sources

* <https://www.youtube.com/watch?v=XrCAvs9AePM&t=955s> – Used for artifact 1
* <https://www.youtube.com/watch?v=mPCZLOVTEc4&t=4s> – Used for artifact 2
* <https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999> - Used for technical discussion
* <https://blog.roboflow.com/yolov5-improvements-and-evaluation/> -Used for getting information on YOLOv5
* <http://vis-www.cs.umass.edu/lfw/> - Used for the positive image dataset
* <https://opencv.org/> - Contains the features used for both Artifact 1 and 2.

FACULTY OF INFORMATION AND

COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as “the unacknowledged use, as one’s own work, of work of another person, whether or not such work has been published” (Regulations Governing Conduct at Examinations, 1997, Regulation 1 (viii), University of Malta).

I the undersigned, declare that the assignment submitted is my work, except where acknowledged and referenced.

I understand that the penalties for making a false declaration may include, but are not limited to, loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

Liam Bugeja Douglas 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student Name Signature

ICS2207 MachineLearning-Project

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Course Code Title of work submitted

20/01/2022

\_\_\_\_\_\_\_\_\_\_\_\_\_

Date