Face Detection and Extraction Using Viola–Jones Algorithm

Mayukh Ghosh, Tathagata Sarkar, Darshan Chokhani, and Anilesh Dey

Abstract In the current times, face detection by computer system has become a major field of interest. Face detection technology can be applied to various fields-including security, biometrics, law enforcement, entertainment, and personal safety—to provide surveillance and tracking of people in real time. Face detection applications use algorithms to find only the human faces within larger images. Face detection algorithms typically start by searching for human eyes, one of the easiest features to detect. The algorithm might then attempt to detect eyebrows, mouth, nose, nostrils and the iris. Once the algorithm concludes that it has found a facial region, it applies additional tests to confirm that it has, in fact, detected a face. In this report, we propose a human face detection method for colored as well as gray images. Also, we cropped the particular detected facial image and extracting and showing the individual cropped image if the input image contains many faces.

Keywords Image processing · Face detection · Viola–Jones algorithm · Cascade object detector · Human faces · Gray images

1 Introduction

Image processing is the tool which is used to process digital images through an algorithm.

Image processing is a multidisciplinary field, with contributions from different branches of science including mathematics, physics, optical, and electrical engineering. Moreover, it overlaps with other areas such as pattern recognition, machine learning, artificial intelligence, and human vision research. Different steps involved in image processing include importing the image with an optical scanner or from a digital camera, analyzing and manipulating the image (data compression, image enhancement, and filtering), and generating the desired output image [1].

M. Ghosh · T. Sarkar · D. Chokhani · A. Dey (✉)
Narula Institute of Technology, Agarpara, India
e-mail: anileshdey@ieee.org

M. Mitra et al. (eds.), *Computational Advancement in Communication, Circuits and Systems*, Lecture Notes in Electrical Engineering 786,
https://doi.org/10.1007/978-981-16-4035-3_9

The generation and development of digital image processing are mainly affected by three factors: first, the development of computers; second, the development of mathematics (especially, the creation and improvement of discrete mathematics theory); third, the demand for a wide range of applications in environment, agriculture, military, industry, and medical science [2–4]. In medicine, it is used for diagnostic imaging modalities such as digital radiography, positron emission tomography (PET), computerized axial tomography (CAT), magnetic resonance imaging (MRI), and functional magnetic resonance imaging (fMRI). Industrial applications include manufacturing systems such as safety systems, quality control, and automated guided vehicle control [12]. Face detection, also known as facial detection, is an artificial intelligence (AI)-based computer technology which is used to find and identify human faces in digital images. Many attempts have been made to solve human face detection problem. The previous trials are aimed for gray-level images only, and image pyramid schemes are necessary to scale with unknown face sizes.

The aim of this project is to successfully detect all the human faces present in a picture and to count the number of faces present in that picture. This algorithm is effective in screening all the non-human faces (dogs, cats, etc.). Such an algorithm which does not involve too many parameters, which is very simple to understand and code, yet giving the desired results with greater efficiency adds to the success of this project.

The scope of this project in the near future is detecting human faces

- in harsh lighting conditions without missing out even a single face,
- even if all the parameters of face detection may not be present (example—a side facing person where one half portion of the face is visible),
- even if the image is blurred,
- if the camera is far away from the subject.

2 Materials and Methods

In this research work, an image containing one or more human faces is given by the user. The code analyzes the image and detects all the human faces present in that image, only when the faces satisfy all the requirements of face detection. In this work, cascade object detector is used which follows Viola–Jones algorithm. Once the detection is completed, all the detected human faces are cropped out from the image and are shown in another output screen. The code also calculates the number of human faces detected in the input image and does not consider any other faces except humans even if they satisfy all the basic requirements of face detection [7].

2.1 Data Used for Image Insertion

The user inputs an image and the MATALB reads it using imread() function. The image can be of jpg, png, jpeg, or any other format. The code also calculates the size of the image using size() function and also resize if width is greater than "320" using imresize() function.

2.2 Data Used for Human Face Detection

For detecting human face, an object called vision.CascadeObjectDetector(_) has been used. This helps us to detect only the human faces with 95% accuracy. Now a function called step() has been used to return the location of the detected regions. For detecting the number of detected region, we use size() function. Then numbers are converted into string using num2str() function. Finally to concatenate the strings, the function strcat() has been used. Now in the output window along with the detected faces, the number of faces detected will also be shown.

Viola–Jones Algorithm Developed in 2001 by Paul Viola and Michael Jones, the Viola–Jones algorithm is an object-recognition framework that allows the detection of image features in real time. Despite being an outdated framework, Viola–Jones is quite powerful, and its application has proven to be exceptionally notable in real-time face detection [5]. Viola–Jones algorithm is used within cascade object detector to detect people's faces, noses, eyes, mouth, or upper body [6]. The Viola–Jones algorithm is a widely used mechanism for object detection. The main property of this algorithm is that training is slow, but detection is fast. This algorithm uses Haar basis feature filters, so it does not use multiplications [12].

The problem to be solved is detection of faces in an image. A human can do this easily, but 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 toward 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 [11].

The characteristics of Viola–Jones algorithm which make it a good detection algorithm are: Robust—very high detection rate (true-positive rate) and very low false-positive rate always. Real time—For practical applications, at least 2 frames per second must be processed.

Face detection only (not recognition)—The goal is to distinguish faces from non-faces (detection is the first step in the recognition process).

The algorithm has four stages:

1. Haar feature selection
2. Creating an integral image

3. AdaBoost training
4. Cascading classifiers.

Haar-Like Features Often in computer vision, features are extracted from input images rather than using their intensities (RGB values, etc.) directly. Haar-like features are one example. Other examples include histogram of oriented gradients (HOG), local binary patterns (LBP), etc. A Haar-like feature consists of dark regions and light regions. It produces a single value by taking the sum of the intensities of the light regions and subtract that by the sum of the intensities of dark regions. There are many different types of Haar-like features, but the Viola–Jones object detection framework only uses the ones in Fig. 1. The different types of Haar-like features let us extract useful information from an image such as edges, straight lines, and diagonal lines that we can use to identify an object (i.e., the human face) [9] (Fig. 2).

Creating an Integral Image An integral image is an intermediate representation of an image where the value for location (x, y) on the integral image equals the sum of the pixels above and to the left (inclusive) of the (x, y) location on the original image [12]. This intermediate representation is essential because it allows for fast calculation of rectangular region. To illustrate, figures shows that the sum of the red region D can be calculated in constant time instead of having to loop through all the pixels in that region. Since the process of extracting Haar-like features involves

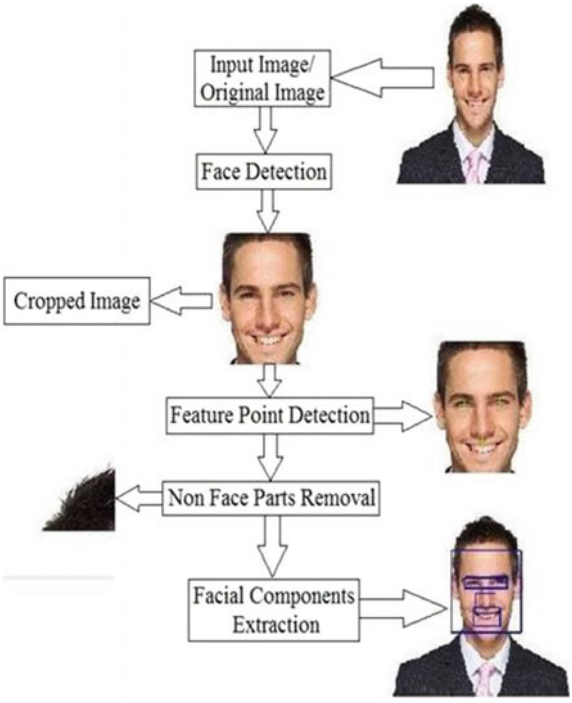


Fig. 1 Working of Viola–Jones algorithm

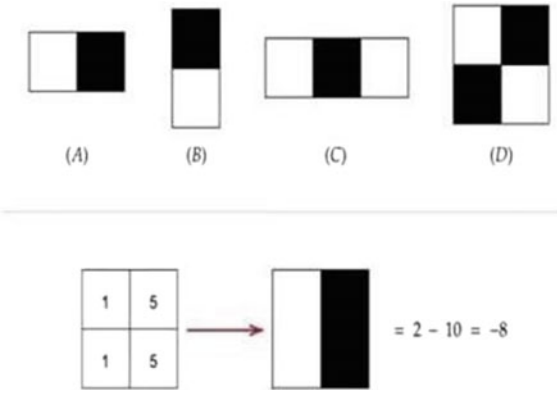


Fig. 2 Haar-like features (top) and how to calculate them (bottom)

calculating the sum of dark/light rectangular regions, the introduction of integral images greatly cuts down the time needed to complete this task [10] (Fig. 3).

$$\theta_j, s_j = \arg\min_{\theta,s} \sum_{i=1}^{N} w_j^i \varepsilon_j^i \tag{1}$$

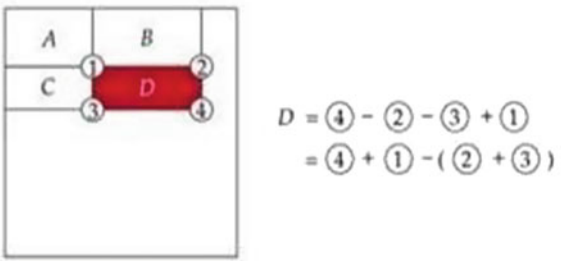


Fig. 3 Conversion of original image into integral image

where,

$$\varepsilon_j^i = \begin{cases} 0 \text{ if } y^i = h_j\left(\mathbf{x}^i, \theta_j, s_j\right) \\ 1 \text{ otherwise} \end{cases} \tag{2}$$

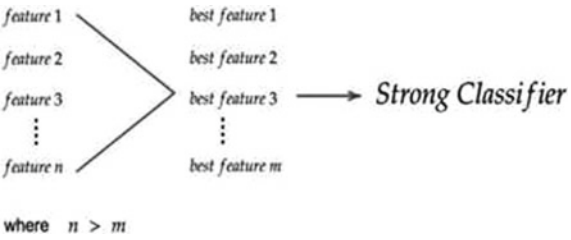
Adaboost Training The AdaBoost (adaptive boosting) algorithm is a machine learning algorithm for selecting the best subset of features among all available features. The output of the algorithm is a classifier (a.k.a prediction function, hypothesis function) called a "strong classifier." A strong classifier is made up of a linear combination of "weak classifiers" (best features). In each iteration, the algorithm finds the error rate for all features and then chooses the feature with the lowest error rate for that iteration [10]. The object detection framework employs a variant of the learning algorithm AdaBoost to both select the best features and to train classifiers that use them. This algorithm constructs a "strong" classifier as a linear combination of weighted simple "weak" classifiers.

$$h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right) \tag{3}$$

Each weak classifier is a threshold function based on the feature f_j.

$$h_j(\mathbf{x}) = \begin{cases} -s_j \text{ if } f_j < \theta_j \\ s_j \quad \text{otherwise} \end{cases} \tag{4}$$

The threshold value θ_j and the polarities $s_j \in \pm 1$ determined in the training, as well as the coefficients a_j.



Cascading Classifiers A cascade classifier is a multi-stage classifier that can perform detection quickly and accurately. Each stage consists of a strong classifier produced by the AdaBoost algorithm. From one stage to another, the number of weak classifiers in a strong classifier increases. An input is evaluated on a sequential (stage by stage) basis. If a classifier for a specific stage outputs a negative result, the input is discarded immediately. In case the output is positive, the input is forwarded onto the next stage [5] (Fig. 4).

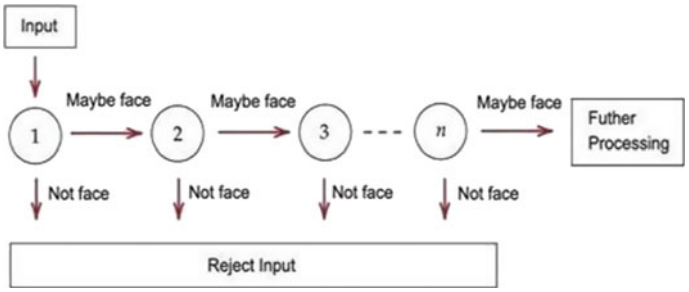


Fig. 4 Working of cascading classifiers

Cascading Basis Cascading is another sort of "hack" to boost the speed and accuracy of our model. So, we start by taking a sub-window, and within this sub-window, we take our most important or best feature and see if it is present in the image within the sub-window. If it is not in the sub-window, then we do not even look at the sub-window, we just discard it. Then if it is present, we look at the second feature in the sub-window. If it is not present, then we reject the sub-window. We go on for the number of features have, and reject the sub-windows without the feature. Evaluations may take split seconds; but since you have to do it for each feature, it could take a lot of time. Cascading speeds up this process a lot, and the machine is able to deliver better results much faster than ever before.

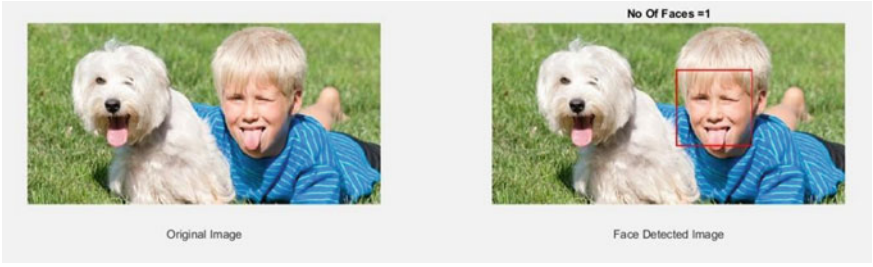
2.3 Datasets

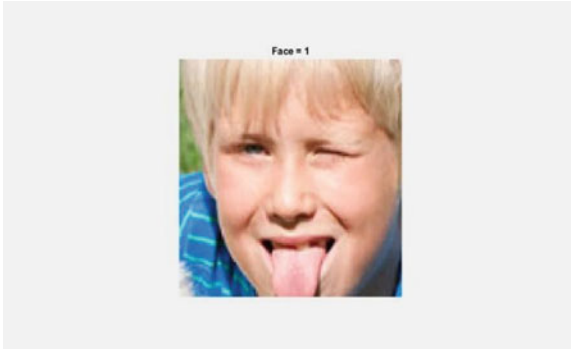
Datasets	Face detection accuracy (%)	Distant image (%)	Non-human face (%)	Blurred image (%)	Side image (%)
[1]	100	0	0	0	0
[2]	87.5	12.5	0	0	0
[3]	0	0	0	100	0
[4]	100	0	50	0	0
[5]	100	0	50	0	0
[6]	100	0	75	0	0
[7]	0	50	50	0	100
[8]	0	0	0	0	100
[9]	66.6	0	0	33.3	0
[10]	100	0	100	0	0
Mean	65.41	6.25	32.5	13.33	20

The following datasets were taken in order to measure the accuracy of the algorithm. A total number of ten datasets were taken out of which five of them gave 100%

accuracy, and the mean of all the accuracies resulted to be 65.41%. The datasets which were taken in random included pictures having only people, people with animal, blurred images as well as people facing sideward out of which if in an image contains only animals, then our accuracy of non-detected human face is 100%.

For example, in the dataset [4] which consists of a boy with a dog lying on a green grassy field in background and harsh light conditions where the algorithm properly extracts the boy's face without considering the dog face and also calculates that only one face is present in the image.



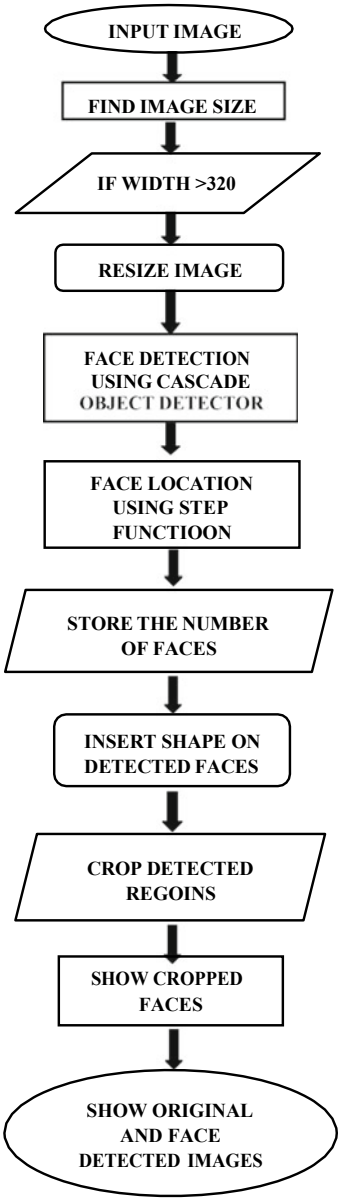


2.4 *Algorithm*

- Read an image using "imread()" function and store in a variable "the_image." (.png /.jpg / jpeg / etc.)
- Store the size of image in a matrix form of order "width × height" using "size()" function.
- Resize the image to the width of "320" using "imresize()" function if width of input image is greater than "320."

- Detect the face using "vision.CascadeObjectDetector()" function and return the location of the detected region of face using "step()" function and store in a variable "location_of_face."
- Calculate the number of extracted faces using "size(location_of_face,1)" function and store in a variable "*n*" and convert it into string using "num2str(n)".
- Insert a shape over the detected faces using "insertShape()" function.
- Crop the detected image using "imcrop()" function.
- Now extract the individual faces from the detected image using a for loop which runs from "1 to *n.*"
- Calculate the location of the individual extracted faces using "location_of_face(i,:)" and crop the respective extracted faces using "imcrop(the_image,predicted_loc)."
- Plot the original image, the face detected image and also show the total number of faces detected.

2.5 Flowchart



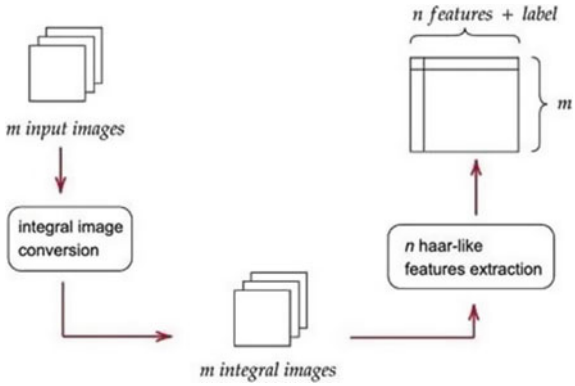


Fig. 5 Data preparation

3 Results and Discussion

3.1 Data Preparation

Assuming that you already have a training set consisting of positive samples (faces) and negative samples (non-faces), the first step is to extract features from those sample images. Viola and Jones [12] recommend the images to be 24 × 24. Since each type of Haar-like features can have different sizes and positions in a 24 × 24 window, over 160,000 Haar-like features can be extracted. Nonetheless, in this stage, all 160,000+ Haar-like features need to be calculated. Fortunately, the introduction of integral images helps speed up this process [10] (Fig. 5).

3.2 Constructing a Cascade Classifier with a Modified AdaBoost Algorithm

As you can imagine, using all 160,000+ features directly are computationally inefficient. Viola and Jones [12] proposed two solutions that can solve this. First, reduce the number of features to only a handful of useful features with the AdaBoost algorithm. Second, split the remaining features into stages and evaluate each input in a stage by stage (cascading) fashion. Viola and Jones [12] devised a modified version of the AdaBoost algorithm to be able to train a cascade classifier [10] (Fig. 6).

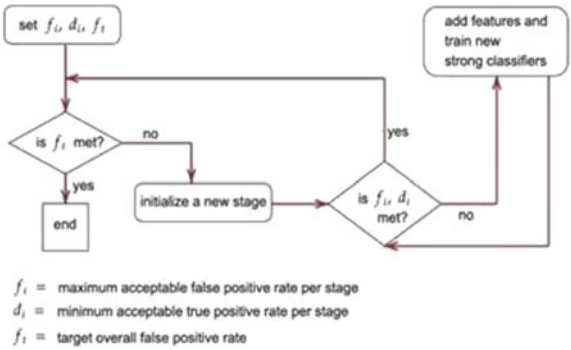
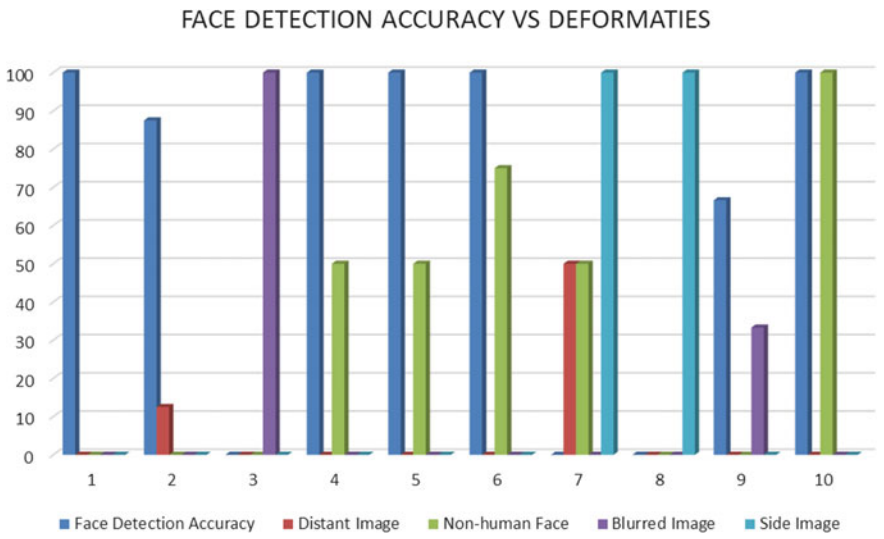


Fig. 6 AdaBoost algorithm

3.3 Statistical Analysis



The graph is based on certain parameters which includes face detection accuracy, distant images, non-human faces, blurred images, and side view images on the *x*-axis, and for the *y*-axis, it contains the percentage. The percentage here is calculated on the basis of the number of elements present in picture as well as the algorithm which is opting here.

For example, the dataset [10] tells us that non-human entity that means either any kind of animal or bird is present in the picture; that is why it is a hundred percent on that aspect as well as the accuracy of the detected face is hundred percent as the algorithm is designed so that it can detect no other face than human faces.

Again, in case of dataset [1], the accuracy of face detected is hundred percent where all the other parameters are zero. This means that it is a proper image of a

person or a group of people having no deformities such as blurred image, non-human face, distant image, or side viewing angle.

In case of dataset [8], the percentage of sideward image is hundred percent, whereas the face detection accuracy is zero percent. For example, consider a picture of a woman holding her baby, both are facing sideward.

For the dataset [4] the percentage of non-human image is fifty percent as only one dog is present, whereas the accuracy of face detection is hundred percent because only one human face is detected. For example, consider a picture of a boy and a dog lying side by side on green field.

3.4 Application

Imagine that we need to detect faces in the above image. Viola and Jones [12] use a sliding window approach where window of various scales is slid across the entire image. The scale factor and the shifting step size are parameters for you to decide upon. So, for the above image, there are m sub-windows to evaluate. For a sub-window "*I,*" the framework resizes the image of that sub-window to a base size of 24 × 24 (to match training data), convert it into an integral image, and feed it through a cascade classifier produced during the training phase. A face is detected if a sub-window passes through all the stages in the cascade classifier (Figs. 7 and 8).

Now, we are also extracting the particular facial images with their respective identification numbers (Fig. 9).

Fig. 7 Input image

Fig. 8 Detected image

Fig. 9 Individually extracted images

4 Conclusion

To conclude, one learns about the Viola–Jones object detection framework and its application for face detection. Many technologies today benefited from Paul Viola and Michael Jones's work. By understanding how the framework works, one can confidently implement their own version of their work or used an open-source implementation like the one provided by OpenCV. It is hoped that the explanation moves someone forward in that direction and compels him/her to create amazing technologies that uses this awesome framework. This research work has various applications in the field of forensic, security, biometrics, law enforcement, entertainment, and personal safety to provide surveillance and tracking of people in real time. Image processing is also used in cameras for detecting the faces for bokeh mode and many such modes. This can also be used in the CCTV cameras present in the school and college classrooms for tracking the number of students present in the class. This is an efficient weapon against bank robberies.

So just to add some close remarks about the Viola–Jones Algorithm:

- The algorithm was developed in 2001 by Paul Viola and Michael Jones, the first of its kind, and was primarily used for facial detection applications.

- There are two steps to the algorithms: there is training with facial and non-facial images, and then there is the actual detection.
- There are two steps for training: training the classifiers and AdaBoost.
- There are two steps for detection: detecting the Haar-like features and creating the integral image.
- Viola–Jones is one of the most powerful algorithms of its time, and even though there are better models out there today, Viola–Jones set the foundation for it in the field of facial detection.

References

1. Rosenfeld A (1969) Picture processing by computer. Academic Press, New York
2. Gonzalez RC (2008) Digital image processing. In: Woods RE (Richard Eugene), 1954, 3rd ed. Prentice Hall, Upper Saddle River, N.J., pp 23–28
3. Chakravorty P (2018) What is a signal? [Lecture Notes]. IEEE Sig Process Mag
4. Gonzalez R (2018) Digital image processing. Pearson, New York, NY
5. Papageorgiou C, Oren M, Poggio T (1998) A general framework for object detection. In: International conference on computer vision
6. Rathore D, Das GK, Saxena A (2020) Concept of face recognition: a review. Int J Technol Res Eng 7(5)
7. Blake C, Keogh E, Merz C (1999) UCI repository of machine learning databases. Department of Information and Computer Science, University of California at Irvine, Irvine, CA
8. Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: Machine learning: proceedings of the thirteenth international conference. Morgan Kauman, San Francisco, pp 148–156
9. Mordvintsev A, Abid K (2013) Face detection using Haar cascades. OpenCV-Python Tutorial
10. Jensen OH (2008) Implementing the Viola-Jones face detection algorithm. PhD thesis, Technical University of Denmark, DTU
11. Ramsri (2012, September 16) Viola-Jones face detection and tracking explained. In: Cen K (2016) Study of Viola-Jones real time face detector
12. Viola P, Jones M (2001) 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, vol 1, pp I-511–I-518. https://doi.org/10.1109/CVPR.2001.990517