**Mini Project Report on**



**FACE DETECTION**



**Submitted in partial fulfilment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Name- EKANSH TYAGI**  **Uni. Roll No- 2016743**

***Under the Mentorship of***

**Mr. SAURABH KUMAR MISHRA**

**Assistant Professor**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**January 2023**

A picture containing text

Description automatically generated

**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“FACE DETECTION”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Mr. SAURABH KUMAR MISHRA, Assistant Professor** , Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Ekansh Tyagi 2016743

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction | 4-5 |
| Chapter 2 | Literature Survey | 6-9 |
| Chapter 3 | Methodology | 10-12 |
| Chapter 4 | Result and Discussion | 13-14 |
| Chapter 5 | Conclusion and Future Work | 15 |
|  | References | 16 |

**Chapter 1**

**Introduction**

1. **PROBLEM STATEMENT-**

**Perform Face Detection using Python and OpenCV.**

In this project, I have developed a code for Face Detection technology using OpenCV libraries in python language with the help of machine learning. This project detects the faces in the given uploaded image and returns the number of faces detected. The basis of this project is the HAAR Cascade Algorithm. I have used the OpenCV dataset for detecting frontal face features inside a photograph. The project is developed on GOOGLE COLAB IDE.

1. **INTRODUCTION-**

Face detection also known as Facial Detection is a computer technology that uses Artificial Intelligence (AI) to find and recognize human faces in digital photographs. Face Detection technology can be used to enable real-time surveillance and tracking of people in a variety of industries, such as SECURITY, BIOMETRICS, LAW ENFORCEMENT,ENTERTAINMENT and PERSONAL SAFETY.

From basic computer vision methods to developments in machine learning (ML) to increasingly complex artificial neural networks(ANN) and related technologies, face detection has advanced, ongoing and leading performance. It now serves as the foundation for a number of crucial applications such as Face Tracking, Face Analysis, and Facial Recognition.

The application’s ability to perform sequential tasks has a substantial impact on face detection.

Face detection aids in determining which areas of a picture or video should be focused on in face analysis in order to identify age, gender and emotions through facial expressions. Face detection is necessary for algorithms that determine which elements of an image or video are required to build a faceprint in a facial recognition system, which maps on individual’s facial characteristics and saves the data as a faceprint.

The algorithm which is the basis of the face detection project is the HAAR CASCADE ALGORITHM. This algorithm can be used to detect faces inside a photograph or a video. This consists of various data sets which are used to detect various body parts of a human inside a photo or a video. The various datasets of the HAAR CASCADE ALGORITHM can be found easily over the internet (most commonly over GitHub). It is a part of the openCV library.

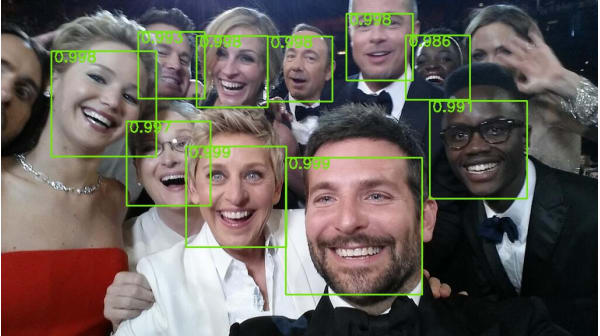


Fig. 1.1 Face Detection Example 1



Fig. 1.2 Face Detection Example 2

The algorithm and code for Face Detection are very easy to understand and implement. And as described before, Face Detection acts as a basis for various other projects like Facial recognition, Face Analysis etc.

This project uses two main libraries, namely OpenCV and numpy. OpenCV is used to perform all the basic detection operations and numpy is used for counting the number of faces.

The Accuracy of this project is not 100%.

**Chapter 2**

**Literature Survey**

**2.1 Abstract-** Due to the complexity of faces as objects and the numerous applications that need face recognition as a first step, face perception is now a study topic of interest in the computer vision literature. This essay aims to provide a critical overview of the body of knowledge on human face detection systems. Face identification is a challenging task in image analysis, which has new applications every day.

The face identification issue may be described as a computer vision challenge that entails identifying one or more human faces in a picture. One of the earliest and most crucial phases in face analysis is this one.

Face identification is challenging due to several picture appearance differences, including position variation (front vs. non-front), occlusion, image orientation, lighting conditions, and face expression. Three widely used face detection techniques are given in this work. Survey and assessment become more crucial when there are more recommended approaches.

**2.2 Introduction-** Several fields, including image processing, pattern recognition, computer vision, neural networks, cognitive science, neurology, psychology, and physiology are actively researching the topic of face identification. It is a distinct procedure, not just a use of object recognition software in general. It also serves as a symbol for the most magnificent aspects of human eyesight.

The foundation of all automatic facial image analysis applications, such as face recognition and verification, face tracking for surveillance, facial behaviour analysis, facial attribute recognition, and gender/age recognition, is automatic face detection.

The purpose of face detection is to identify whether there are any faces in the image and, if so, to return the location and size of each face in the image. For humans, this may seem like a simple task, but for computers, it is incredibly difficult. Over the past few decades, this has been one of the most widely explored study areas.

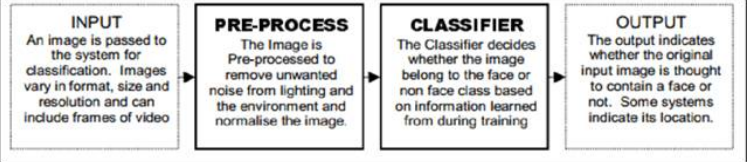


Fig 2.1 Basic Face Detection System

**2.3 Methods of Face Detection-**

1. **Skin Color Based Face Detection-** Skin tone recognition is a fairly common and effective method for face recognition in colour photographs. The colour of a person's face is a significant aspect. There are various benefits to monitoring a face using skin tone as a characteristic.Processing of colours occurs considerably more quickly than processing of other facial aspects. Each pixel was categorised as skin or non-skin throughout the skin colour identification procedure based on its colour components.



Fig. 2.2 Skin Colour Based Face Detection

1. **Viola Jones Face Detection System-** Paul Paul Viola and Michael Jones presented the Viola-Jones object detection framework in 2001 as an object detection framework that offers reliable and competitive object identification rates in real-time. It was primarily driven by the job of face identification, despite the fact that it can be trained to recognise a range of object classes. This face detection framework has the speed and accuracy to process photos at high detection rates. The architecture for face detection has three key phases.

**1. Integral Picture**: This novel image format enables very quick computation of the detector's feature set. Harr-like features may be computed at any size or location in constant time after the integral picture has been calculated.

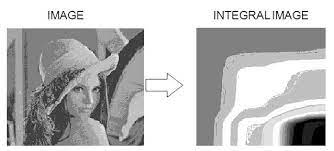
****

Fig. 2.3 Conversion of image into integral image

**2. Adaboost Algorithm:** At this step, classifiers are built by choosing a few crucial characteristics (rectangle features) with the aid of the Adaboost algorithm. We are only interested in a small subset of the many characteristics calculated in stage 1 that would allow us to recognise faces quite accurately. We employ the Adaboost Algorithm for this.  To choose key traits and develop classifiers that would employ them. This algorithm's goal is to build a powerful classifier by linearly combining weak classifiers. AdaBoost offers a powerful algorithm for learning.

**C. Cascading:** By concentrating attention on potential face-like portions of the picture, the third main stage of this technique greatly boosts the speed of the detector. This is done by merging progressively more complicated classifiers in a cascade structure. The classifiers in this cascade structure work together. It operates in a way where simpler initial classifiers are used to reject the majority of sub-windows and more complicated classifiers are then used to achieve low false positive rates.

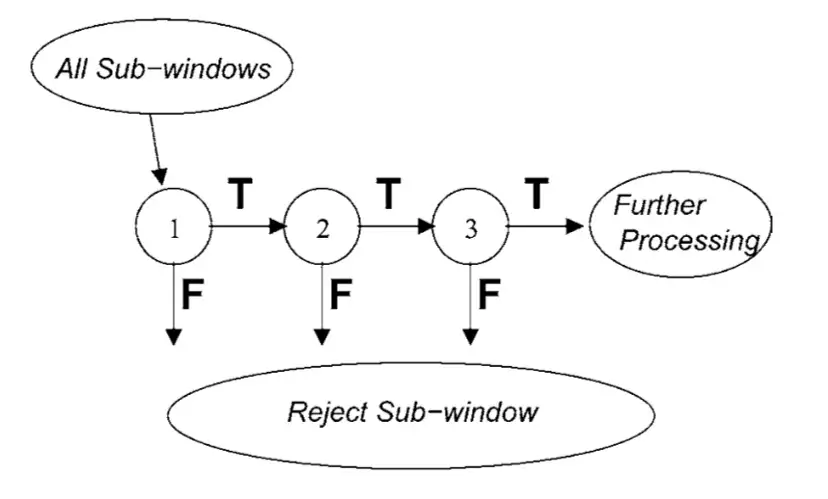


Fig. 2.4 Cascading

**2.4 Conclusion of Literature Survey-**

We learn about the difficulties in face detection by referring to several approaches. We have concluded from this literature review that it is crucial to exclude background information. Face detection would be improved by eliminating unnecessary information such noise, non-face parts, and backdrop.

We also conclude that following things make face detection more complicated.

1. Different Facial poses
2. Complex background
3. Varied facial expression
4. Overlapping Faces

Based on skin tone, face identification algorithms may be employed effectively in photographs with either a basic or complicated background. All of the faces in the photographs can be accurately identified by the algorithm

A face identification method with excellent detection accuracy and little computing time was described by Viola Jones. The method's Haar-like properties are relatively straightforward and efficient for frontal face identification, but they are less suitable for faces in arbitrary positions.

**Chapter 3**

**Methodology**

This project is developed on GOOGLE COLAB IDE.

First step is as always including the libraries.

The two libraries used for this project are:-

1. OpenCV- This library allows us to perform image processing and computer vision tasks.
2. NumPy- It is a library for the python language which adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

The algorithm which is the basis of this project is known as the HAAR CASCADES ALGORITHM. Also known as the Viola-Jones Face Detection Technique, this algorithm was discovered long before deep learning came into popularity. This algorithms is still found to be used almost everywhere. Its fully trained models are available on OpenCV GitHub Repository.

*“Basically, This algorithm is an Object Detection Algorithms used to identify faces in an image or a real time video. The edge or line detection characteristics that Viola and Jones suggested in their 2001 study “Rapid Object Detetction using a Boosted Cascade of Simple Features” are used by the technique. To train the algorithm, it is given a large number of phots with faces and a large number of negative images without any faces.”*

In the next step, a Cascade Classifier object will be created. The path to the classifier file is sent as an argument to the function object () of this class. We will apply the classifier file haarcascade\_frontalface\_default.xml in this case (A data set from the OpenCV library to detect the facial features inside a photograph or a video). I have used the entire file path to ensure that the file is accessed properly.

Now, the imread function of the cv2 module will then be used to load the picture. The path to the picture will be provided as input to this function. We will use the image we wish to use for face detection as our input.

In order to apply the classification, the image will be converted to grayscale. We use the CvtColor funvtion to do this. This will take the original image as input and the code for the colour space conversion as the second input. In this instance, we utilize the COLOR BGR2GRAY function to convert RGB to Grey.

Now we will use the detetctMultiscale method on our face cascade object to perform the real face detection. We will provide our transformed grayscale image as input. Additionally, various tuning options may be sent to this method, such as scaleFactor or the object to detect minimum size. For the sake of simplicity, I will just pass the image as input.

The identified items will be returned by their functions as rectangles, so we an easily mark them in the output image.

The faces in our initial image will have rectangles around them when we iterate the findings.. Calling the rectangle method on the cv2 Module does this. The picture, the beginning and final vertexes of the rectangle, the RGB colour of the rectangle, and the thickness of the rectangle are all inputs for this function. In order to find and draw all the rectangles, we will accomplish this within a for in loop.

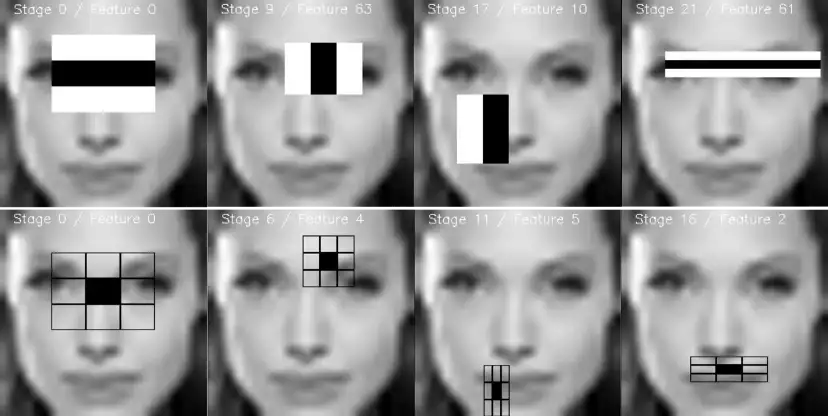


Fig. 3.1 Greyscale image and detecting facial features

Additionally, we display the number of faces in our image in the lower left corner. Given that we are unsure of the colour of the photos, we will first construct another rectangle to place the text on top of.

The starting vertex will be at coordinates (x = 0, y = image height – 25) and the end vertex will be at (x = 270, image height). To avoid more complex code, this is hardcoded for the size of the text we will specify.

Keeping in mind that we can use the same approach we used with the ndarray before to get the image’s height. We use the shape approach to get the number of rows because the image is a matrix.

We will instruct the rectangle function to fill the rectangle by passing it the color white and the thickness -1. The bottom left corner of the picture, the text, the font, the scale of the text, the colour, and the thickness are passed as input to the putText function when writing the text.

Finally, we show the edited image with rectangles and the text.

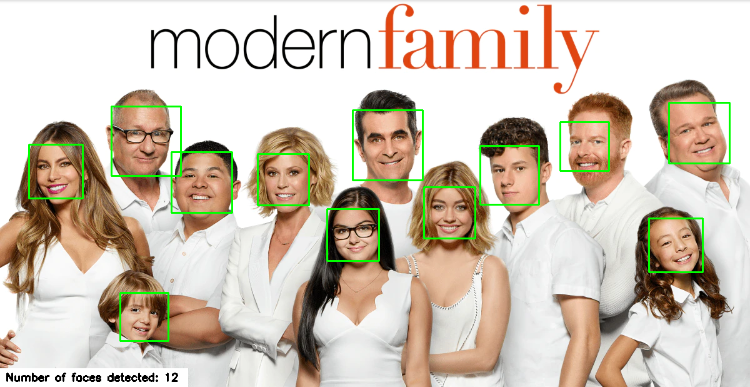


Fig. 3.2 Rectangles are formed around the faces and the number of faces detected is displayed in a separate rectangle.

NOTE- To display the image we have used the cv2\_imshow() function as cv2.imshow() is a disable function in Google Colab.

**Chapter 4**

**Result and Discussion**



Fig 4.1 Resulting Image

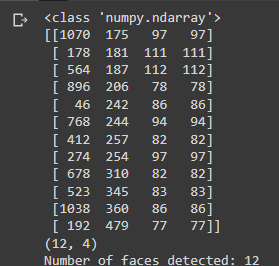


Fig 4.2 Resulting Array

The detected faces are displayed with a rectangle around them and the number of faces are displayed with the help of another rectangle. The numpy array is also displayed in order to count the number of faces.

As mentioned before this program does not give 100% accurate results. Sometimes it can give random rectangles at places where there is no face. I have checked this by performing face detection on multiple images. Sometimes it also doesn’t identify a face.

Although, the accuracy can be increased by making subtle changes into the dataset.

**Chapter 5**

**Conclusion and Future Work**

**5.1 CONCLUSION-**

In this project, I have made a very fast and highly accurate face detection application. We also learnt about the HAAR CASCADES algorithm through this project. This algo acts as the basis for Face Detection and its various applications. The many training models of this algorithm can be accessed through OpenCV github repository.

We also learnt about various cv2 functions within pyhton and also learnt to handle various errors in our code. We also perfected our skills to learn Google Colab IDE. Other IDEs which can be used to perform this project are PyCharm, Jupyter Notebook or Kaggle etc.

The main job within this project is of the haarcascade\_frontalface\_default.xml training model. This model stores the default features of a human face and is used to detect them within a photo or a video. This model was downloaded from GitHub.

Through this project we also learnt about two python libraries namely, OpenCV and NumPy and their various features.

* 1. **FUTURE WORK-**

This project can have various types of usage in the future: -

* **Improved Security-** Face detection improves surveillance efforts and helps track down criminals and terrorists. Personal security is also enhanced since there is nothing for hackers to steal or change, such as passwords.
* **Easy to Integrate-** Face detection is easy to integrate, and most solutions are compatible with most of the security software.
* **Automated Identification-** In the past, identification was manually performed by a person; this was inefficient and frequently inaccurate. Face detection allows the identification process to be automated, thus saving time and increasing accuracy.

**References-**

[1] Binesh T.R and P. Simon, “Fast Pedestrian Detection using smart ROI separation and Integral image based Feature Extraction”, International Journal on Computer Science and Engineering (IJCSE), Vol.4, No.11,pp.1771-1779, 2012.

[2] H.A. Rowley, S. Baluja, and T. Kanade, “Neural Networks Based Face Detection”, IEEE Trans. Pattern Analysis an Machine Intelligence, vol. 20, no. 1, pp. 22-38, 1998.

[3] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features”,Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR , Vol.1, pp.I-511 - I-518, 2001.

[4] <https://www.techtarget.com/searchenterpriseai/definition/face-detection>

Website- www.techtarget.com