To study Detecting and Recognizing Faces
Date of Performance: 21/08/2023
Date of Submission: 28/08/2023

Aim: To study Detecting and Recognizing Faces

Objective: To Conceptualizing Haar Cascades Getting Haar cascade data Using Open CV to Perform face detections performing face detection on still images

Theory:

Conceptualizing Haar Cascades

Getting Haar Cascade Data

Using Open CV to perform Face Detection:

Performing Face detection on a still image:

Introduction

Discover object detection with the Haar Cascade algorithm using OpenCV. Learn how to employ this classic method for detecting objects in images and videos. Explore the underlying principles, step-by-step implementation, and real-world applications. From facial recognition to vehicle detection, grasp the essence of Haar Cascade and OpenCV's role in revolutionizing computer vision. Whether you're a novice or an expert, this article will equip you with the skills to harness the potential of object detection in your projects.
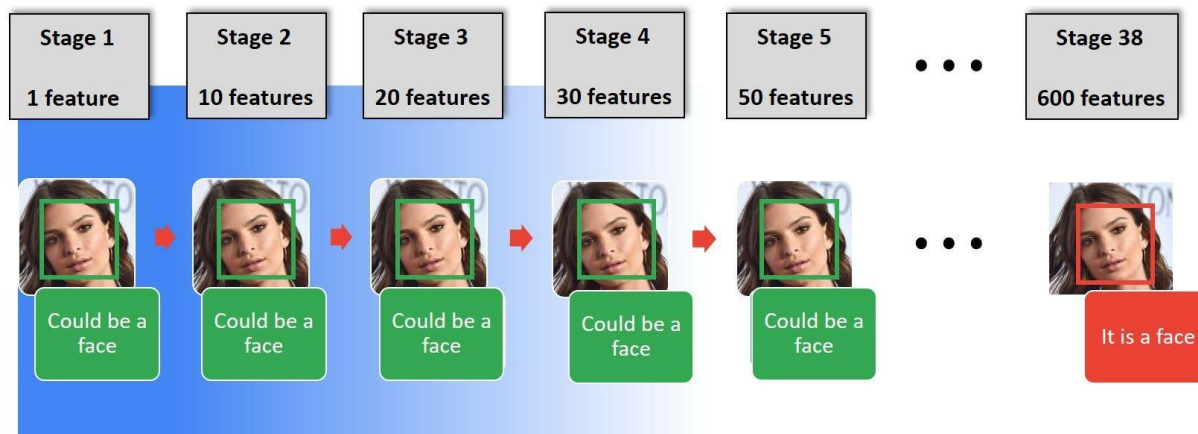

Table of contents

Why Use Haar Cascade Algorithm for Object Detection?

Identifying a custom object in an image is known as object detection. This task can be done using several techniques, but we will use the haar cascade, the simplest method to perform object detection in this article.

What is Haar Cascade Algorithm?

Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location.

This algorithm is not so complex and can run in real-time. We can train a haar-cascade detector to detect various objects like cars, bikes, buildings, fruits, etc.

Haar cascade uses the cascading window, and it tries to compute features in every window and classify whether it could be an object.
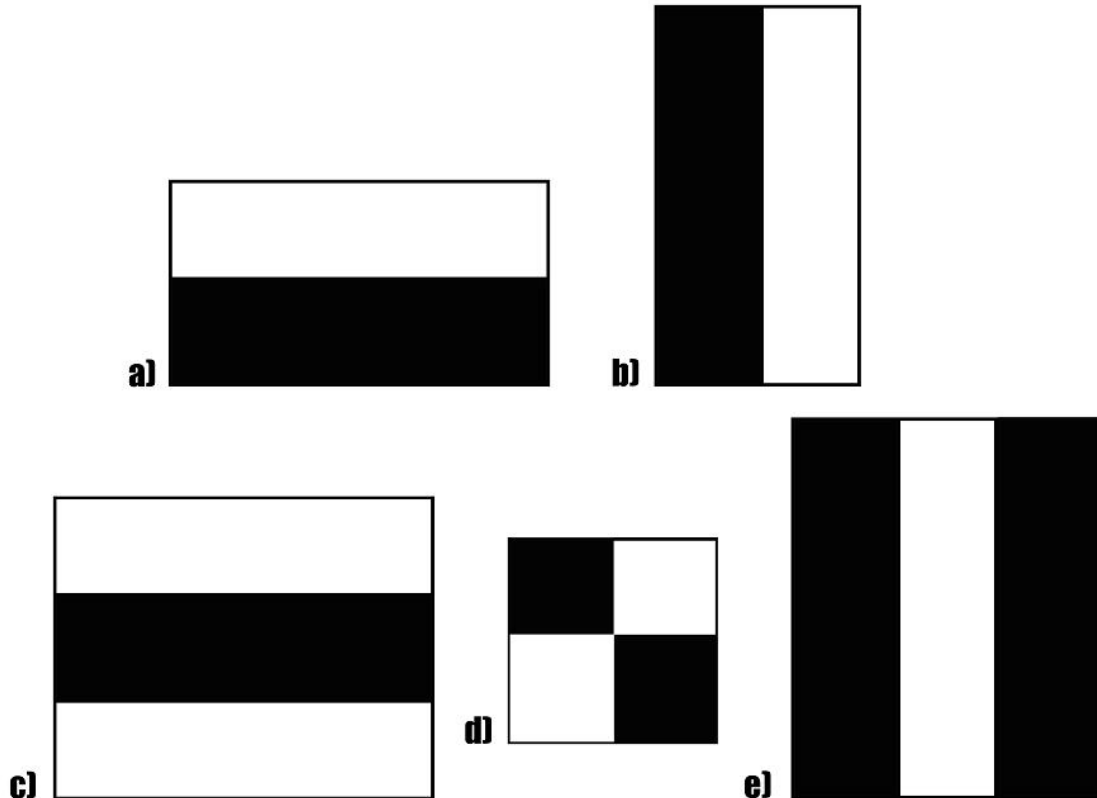
Haar cascade works as a classifier. It classifies positive data points → that are part of our detected object and negative data points → that don't contain our object.

- Haar cascades are fast and can work well in real-time.
- Haar cascade is not as accurate as modern object detection techniques are.
- Haar cascade has a downside. It predicts many false positives.
- Simple to implement, less computing power required.

Code:-

```python
import numpy as np

import cv2

from google.colab.patches import cv2_imshow

face_classifer = cv2.CascadeClassifier('/content/haarcascade_frontalface_default.xml')

color_imgae = cv2.imread('/content/G20_111.jpg')

gray_image = cv2.cvtColor(color_imgae,cv2.COLOR_BGR2GRAY)

faces = face_classifer.detectMultiScale(gray_image,1.3,5)

if faces is ():

  print("No Face Found")

for (x,y,w,h) in faces:

  cv2.rectangle(color_imgae,(x,y),(x+w,y+h),(0,0,0),2)

  cv2_imshow(color_imgae)


cv2.destroyAllWindows()
```

Output:-

Input Image -

Output Image -

Conclusion: In conclusion, the aim of studying the detection and recognition of faces was successfully accomplished. Through this study, we explored various methods and technologies for accurately detecting and recognizing faces within images. Our findings demonstrate the importance of this research in applications such as security, surveillance, and human-computer interaction.