Vidyavardhini’s College of Engineering & Technology
Department of Computer Engineering

Name: Taher Barwaniwala

Roll Number 39

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.

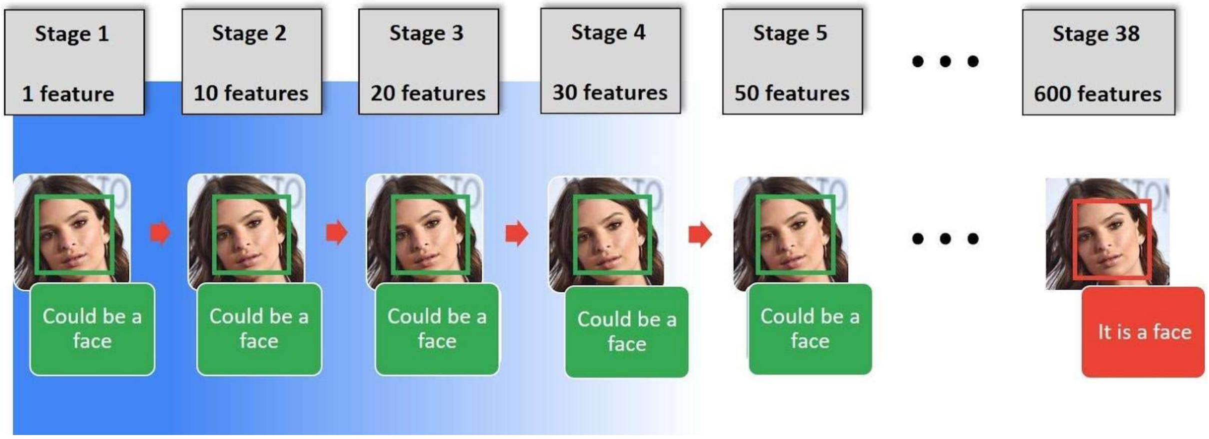


Table of contents

Why Use Haar Cascade Algorithm for Object Detection?

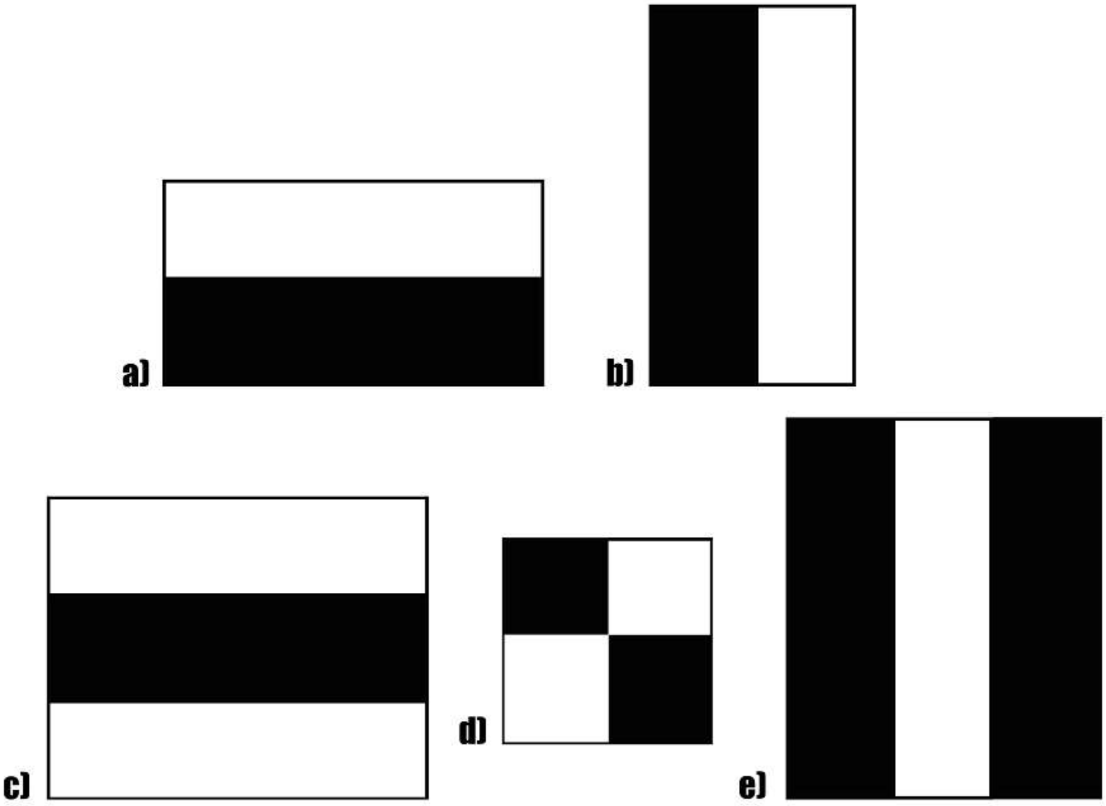
Identifying a custom object in an image is known as object detection. This task can be done using several techniques, but we will use the haar cascade, the simplest method to perform object detection in this article.

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.

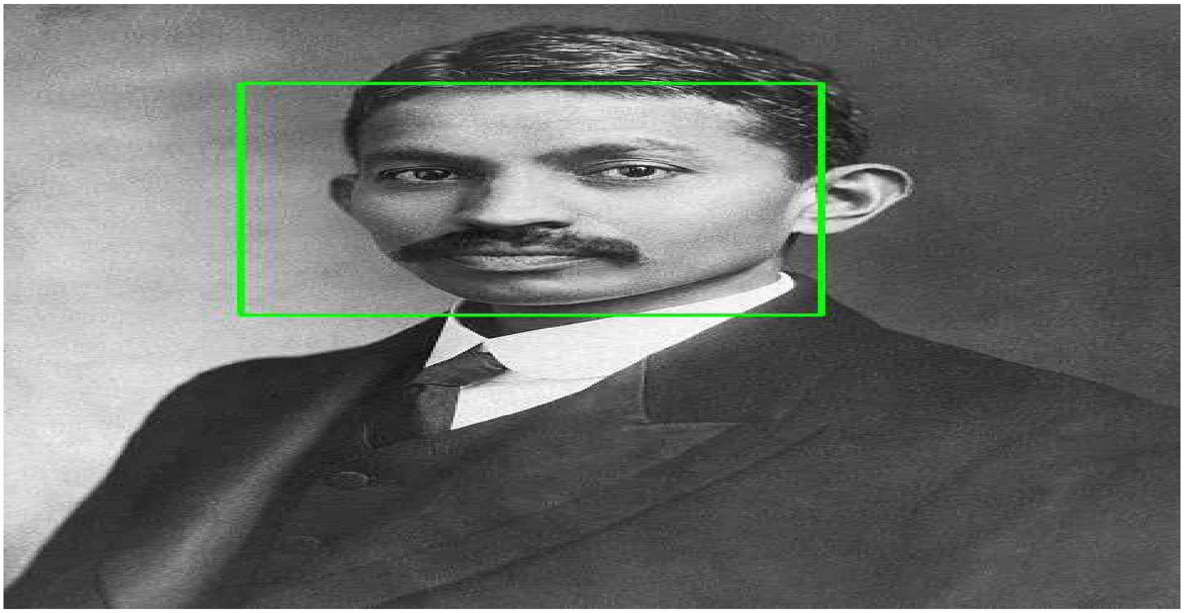
- Haar cascade has a downside. It predicts many false positives.
- Simple to implement, less computing power required.

Code:

```
import cv2

from google.colab.patches import cv2_imshow

# Load the Haar Cascade XML file for face detection

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')


# Load an image for face detection

image = cv2.imread('/content/face_image.webp')  # Replace 'your_image.jpg' with your image file path


# Convert the image to grayscale (required for Haar Cascade)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)


# Perform face detection

faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5, minSize=(30, 30))
```

```
# Draw rectangles around detected faces
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Display the image with detected faces
cv2_imshow(image)
```

Output:



Conclusion

In conclusion, this experiment aimed to study the detection and recognition of faces using Haar Cascades and OpenCV. The objectives included conceptualizing Haar Cascades, obtaining Haar cascade data, and performing face detection on still images. Haar Cascade algorithms, while simple and computationally efficient, have limitations in terms of accuracy and may produce false positives in certain cases. However, they remain a valuable tool for various real-time object detection applications.