Aim: To study Detecting and Recognizing Faces

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

Theory:

Conceptualizing Haar Cascades

The haar computation is performed by comparing the average pixel values at the lighter and darker regions, and then calculating the difference. The haar feature will detect an edge if the difference is near to 1.

Getting Haar Cascade Data

The goal here is to calculate the total of all image pixels in the darker part of the haar feature and the sum of all image pixels in the brighter portion of the haar feature. Then discover their differences. If the image contains an edge dividing dark pixels on the right from light pixels on the left, the haar value will be closer to 1. That is, if the haar value is close to 1, we claim there is an edge identified. There is no edge in the above example because the haar value is distant from 1.

Using OpenCV to perform Face Detection:

1. Install OpenCV
2. Import OpenCV
3. Load the Haar Cascade Classifier
4. Load an Image or Capture from Webcam
5. Perform Face Detection
6. Draw Rectangles Around Detected Faces
7. Display the Result

Performing Face detection on a still image:

1. Import the OpenCV Package
2. Read the Image
3. Convert the Image to Grayscale
4. Load the Classifier
5. Perform the Face Detection

6. Drawing a Bounding Box
7. Displaying the 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.

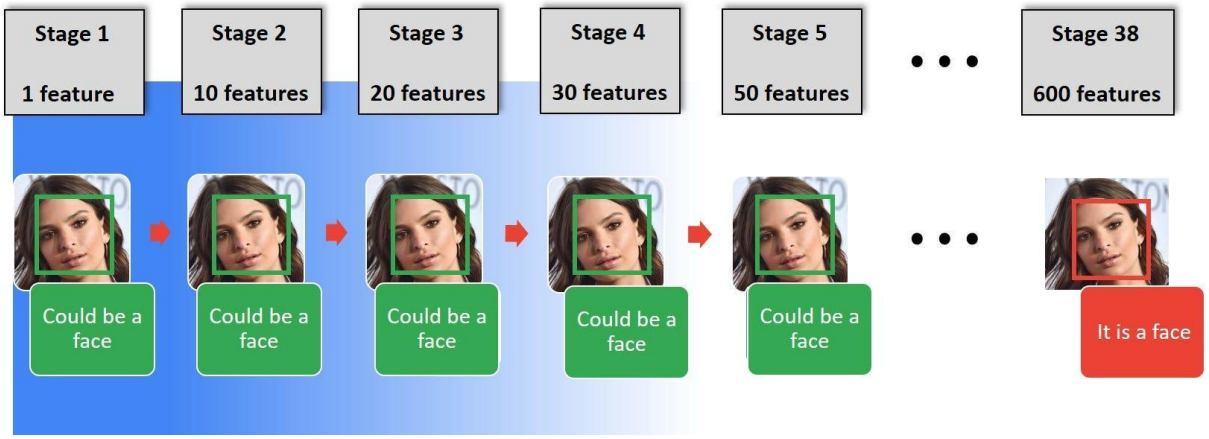


Table of contents

Why Use Haar Cascade Algorithm for Object Detection?

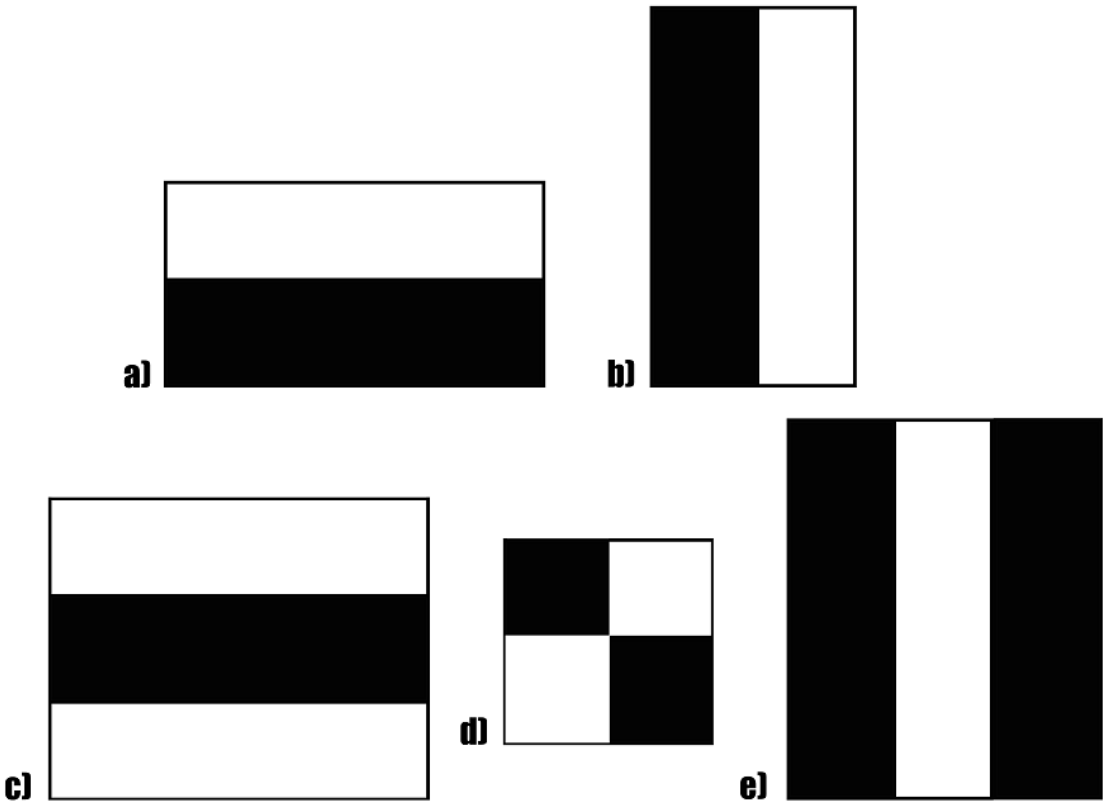
Identifying a custom object in an image is known as object detection. This task can be done using several techniques, but we will use the haar cascade, the simplest method to perform object detection in this article.

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 dlib

import cv2

from google.colab.patches import cv2_imshow

detector = dlib.get_frontal_face_detector()
```

```
input_image = cv2.imread('/content/image.jpg')

cv2_imshow(input_image)

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

faces = detector(gray)

for face in faces:

  x, y, w, h = face.left(), face.top(), face.width(), face.height()

  cv2.rectangle(input_image, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2_imshow(input_image)

print("Number of faces detected:", len(faces))
```

Output:

Conclusion

In short, the Haar Cascade face recognition method is a popular and effective way to find faces in pictures and videos. To recognize facial features, it uses a cascade of straightforward classifiers and pre-trained classifiers. Although it is quick and efficient for real-time applications, it might not be able to handle changes in pose, lighting, and occlusions.

However, Haar Cascade face detection continues to be a useful tool for a variety of real-world applications, particularly when real-time efficiency is important. For the purpose of identifying and detecting faces, OpenCV is a flexible and effective toolkit. For academics and developers working in this field, its versatility, performance, and active community support make it a significant asset.