Name : Ajay Shitkar

Roll No: 14

Branch : Computer Engineering.

Sub : Machine Vision.

Expt : 06

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

Haar Cascades are a machine learning technique for object detection, known for their efficiency. They use simple rectangular features to represent object characteristics and cascade them into a hierarchical structure. This cascade structure rapidly rejects non-object regions, making detection faster. Although effective for tasks like face and pedestrian detection, they have been largely surpassed by deep learning-based approaches for more complex and versatile object recognition tasks.

Using Open CV to perform Face Detection

Using OpenCV for face detection involves utilizing pre-trained Haar Cascade classifiers or deep learning-based models like the Single Shot MultiBox Detector (SSD) or Faster R-CNN. OpenCV provides easy-to-use functions to load these models and apply them to images or video streams. You can fine-tune the detection parameters to balance accuracy and speed. OpenCV's face detection capabilities make it a powerful tool for various computer vision applications.

Performing Face detection on a still image:

Performing face detection on a still image using OpenCV involves loading the image, selecting a pre-trained face detection model (e.g., Haar Cascade or deep learning-based model), and applying the model to locate faces. OpenCV provides user-friendly functions to accomplish this task efficiently. Once detected, faces can be highlighted with bounding boxes, allowing for further analysis or applications like identity recognition, sentiment analysis, or object tracking. This process is a fundamental step in many computer vision and image processing applications.

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.

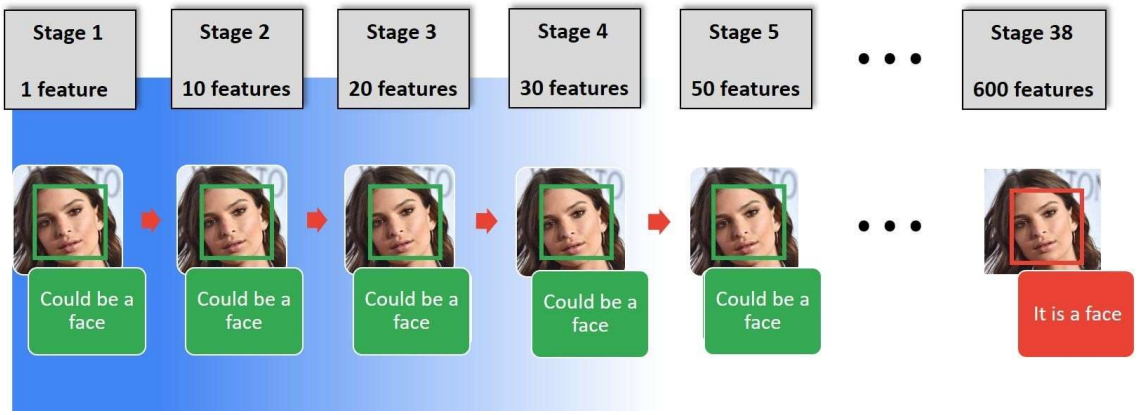


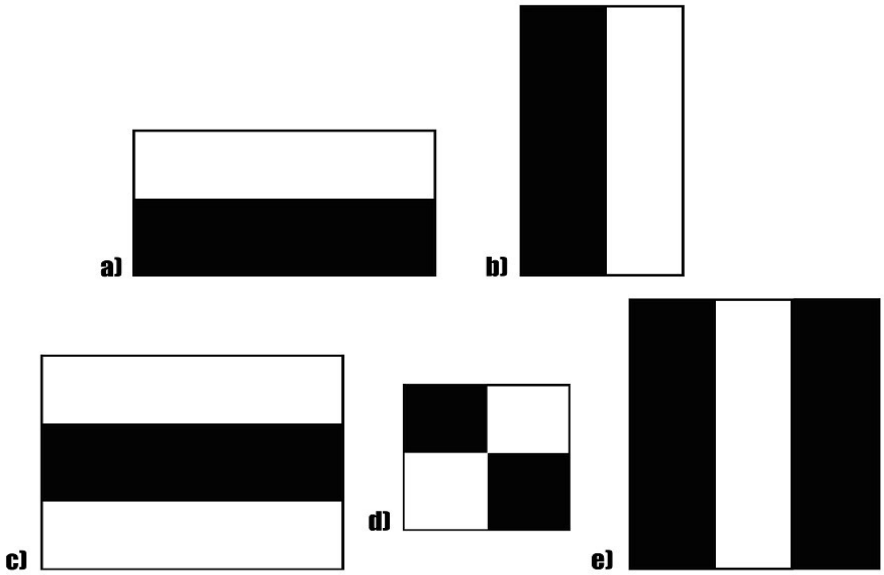
Table of contents

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 haarcascade 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.

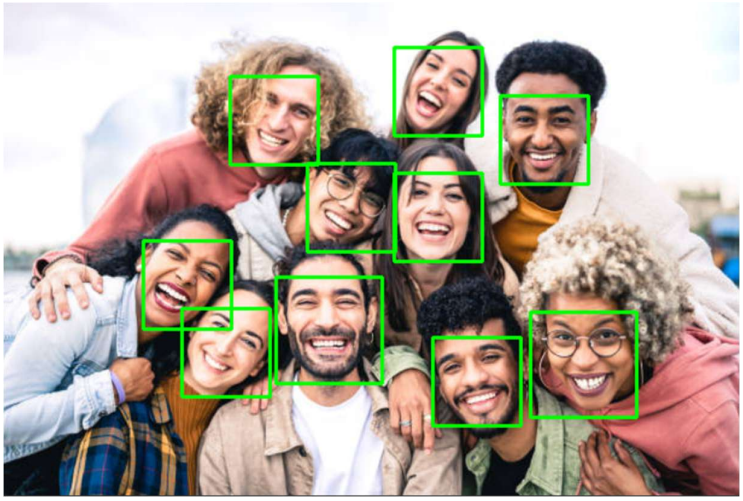
Code:

```
import dlib
import cv2
from google.colab.patches import cv2_imshow

# Load the pre-trained face detection model from dlib
detector = dlib.get_frontal_face_detector()

# Load an image or capture it from a camera
# Replace 'your_image.jpg' with the path to your image or use 0 for the default
camera (webcam)
input_image = cv2.imread('image1.jpg')
cv2_imshow(input_image)

# Convert the image to grayscale for face detection
gray = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)

# Detect faces in the image
faces = detector(gray)

# Draw rectangles around detected faces
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)

# Display the image with faces highlighted
cv2_imshow(input_image)
```

Output:

(Input Image)

(Output Image)



Conclusion :

Haar Cascades represent a classic and efficient method for face detection in computer vision. Obtaining Haar cascade data involves training classifiers on positive and negative examples of the target object, which in this case is faces. OpenCV provides convenient tools for loading pre-trained Haar cascade models. Performing face detection on still images using OpenCV entails applying these models to locate faces within the images. This process serves as a foundational step in various applications, enabling further analysis and tasks like facial recognition, sentiment analysis, or object tracking, making it a valuable tool in the field of computer vision.