

Experiment no 6:

Name - Parth Gharat

Roll No-23

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 method in computer vision for object detection. They use simple Haar-like features to distinguish objects from the background. To create a cascade, a dataset of positive and negative images is needed for training. Integral images speed up feature calculations. Adaboost trains a strong classifier from weak ones based on Haar features. The cascade consists of stages that filter out non-object regions, improving detection speed. Sliding window detection applies the cascade to different positions and scales in an image. Thresholding determines if a region passes each stage. Haar cascades are effective for object detection, often used for tasks like face or pedestrian detection.

Getting Haar Cascade Data :

Obtaining Haar Cascade data for object detection involves either using pre-trained classifiers or creating custom ones:

- **Pre-trained Classifiers:** Download pre-trained Haar Cascade classifiers from sources like OpenCV, GitHub repositories, or online platforms for objects like faces. These are readily available and suitable for common objects.
- **Custom Classifiers:** For unique objects or high precision, create custom Haar Cascade classifiers. Collect positive and negative images, annotate positive images with object bounding boxes, and use tools like OpenCV's

`opencv traincascade` to train the classifier. This process requires careful parameter tuning and substantial computational resources.

Using Open CV to perform Face Detection:

To perform face detection using OpenCV:

- Install OpenCV with `pip install opencv-python`.
- Import OpenCV with `import cv2`
- Load the Haar Cascade classifier for faces.
- Read an image or capture video from a webcam.
- Convert the image to grayscale.
- Use `detectMultiScale` to detect faces, specifying parameters like `scaleFactor`, `minNeighbors`, and `minsize`.
- Draw rectangles around detected faces.
- Display or save the result.
- Release resources if using webcam capture.
- This process enables you to detect and visually highlight faces in images or video streams using OpenCV's pre-trained classifier.

Performing Face detection on a still image:

To perform face detection on a still image using OpenCV:

- Install OpenCV with `pip install opencv-python`.
- Import OpenCV using `import cv2`.
- Load the pre-trained Haar Cascade classifier for faces.
- Read the image you want to analyze.
- Convert the image to grayscale for better detection.

Use the `detectMultiScale` method to identify faces, adjusting parameters like `scaleFactor`, `minNeighbors`, and `minsize` as needed.

Draw rectangles around the detected faces on the original image. Display the result or save it to a file.

If displaying, wait for a key press and then close the OpenCV window.
Execute your script to detect faces in the 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.

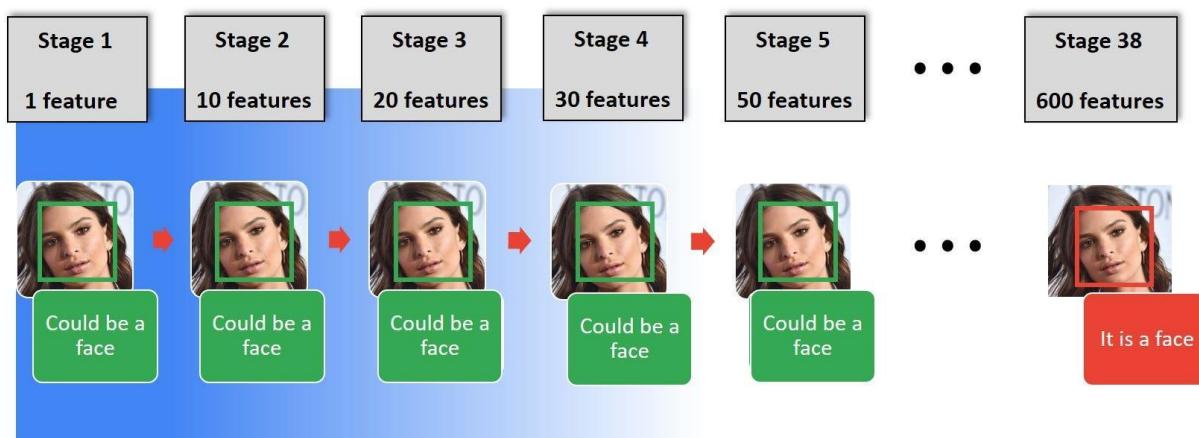


Table of contents

Why Use Haar Cascade Algorithm for Object Detection?

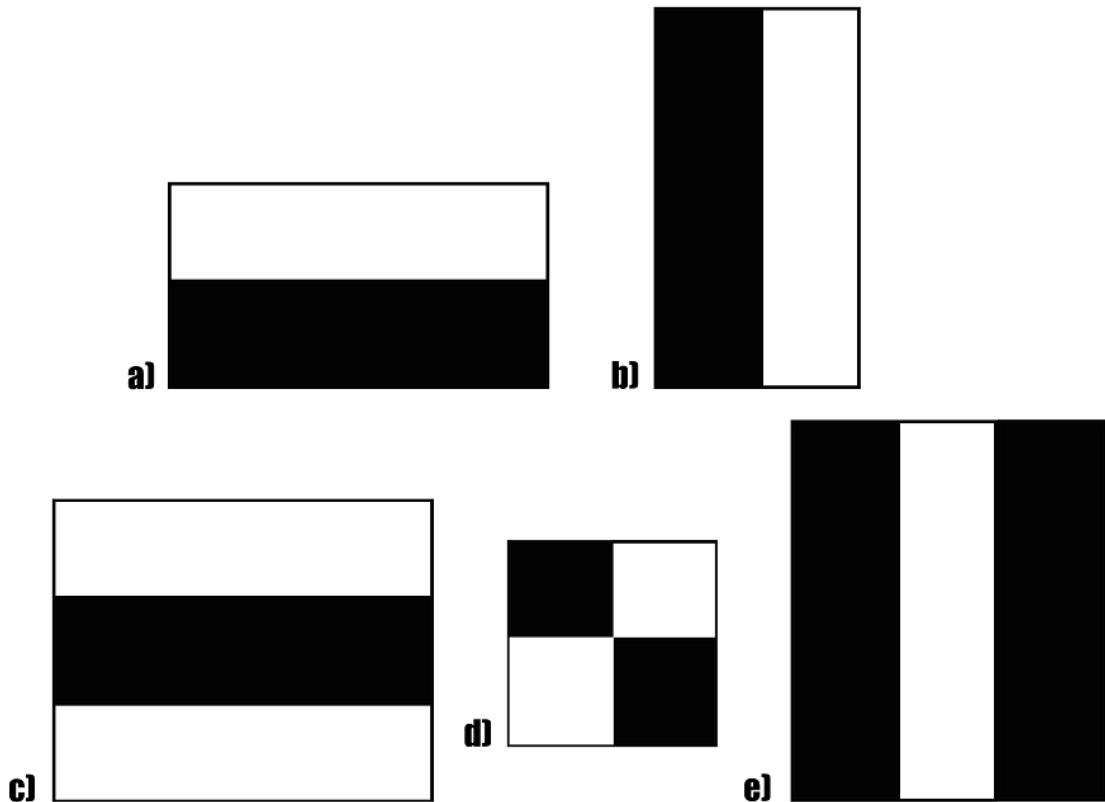
Identifying a custom object in an image is known as object detection. This task can be done using several techniques, but we will use the haar cascade, the simplest method to perform object detection in this article.

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:

```
from google.colab.patches import cv2_imshow

import cv2

import sys


cascPath = "/content/haarcascade_frontalface_default.xml"

faceCascade = cv2.CascadeClassifier(cascPath)

image = cv2.imread("/content/str11.jpg")

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

faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(1,1),
    flags = cv2.CASCADE_SCALE_IMAGE
)

print("Detected {} faces!".format(len(faces)))

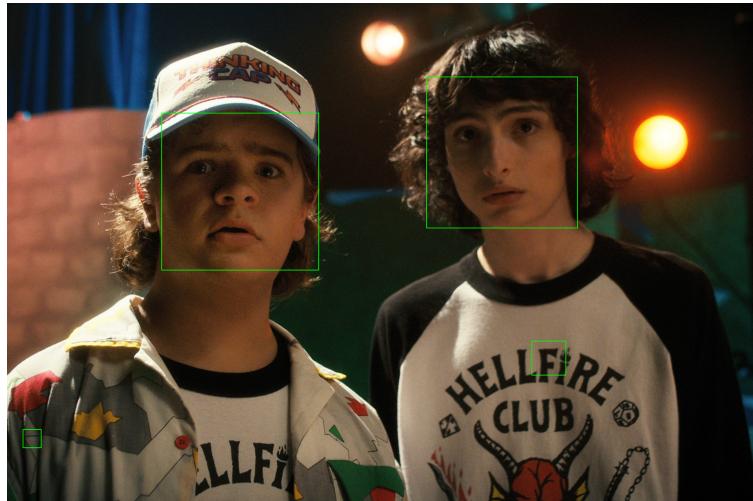
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2_imshow(image)

cv2.waitKey(0)
```



Output:



Conclusion:

- In conclusion, using the Haar Cascade method for face detection in still images proves to be a robust and efficient technique in computer vision. This approach leverages pre-trained classifiers and a straightforward process to accurately identify faces within static images.
- By converting images to grayscale, detecting faces through the `detectMultiScale` function with adjustable parameters, and visualizing the results with bounding rectangles, it provides a reliable foundation for various applications, including image analysis, facial recognition, and object tracking.
- While Haar Cascade-based face detection is effective, it may require parameter fine-tuning for optimal results in specific scenarios, but overall, it remains a valuable tool in the field of computer vision.