Experiment no 6:

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 :-

Conceptualizing Haar Cascades involves understanding the fundamental principles of the Haar Cascade algorithm, which is the cornerstone of object detection in computer vision. This algorithm's key concept revolves around evaluating image regions using a set of features to classify them as containing or not containing a target object. Haar Cascades are known for their scalability, real-time performance, and simplicity, making them a valuable tool for object detection tasks.

Getting Haar Cascade Data:-

Getting Haar Cascade data is a crucial step in implementing the Haar Cascade algorithm for object detection. Haar Cascade data consists of pre-trained classifiers that can recognize specific objects or patterns. Here's an overview of the process:

1. **Data Collection:** First, you need to gather a dataset containing positive and negative images of the object you want to detect. Positive images should contain instances of the object, while negative images should not.

2. **Data Annotation:** Annotate the positive images to mark the regions where the target object is present. This step helps the algorithm learn what the object looks like.

3. **Training:** Use the collected and annotated data to train the Haar Cascade classifier. This process involves computing a set of features (Haar-like features) for each image and training the classifier to distinguish between positive and negative samples.

4. **XML File Generation:** After successful training, the Haar Cascade classifier generates an XML file containing the learned model. This XML file encapsulates the information needed for object detection.

5. **Integration with OpenCV:** You can now use the generated XML file with OpenCV's Haar Cascade functions to detect the object in new images or videos.

Getting Haar Cascade data is a critical step to ensure that your object detection model is capable of recognizing specific objects or patterns effectively. It allows you to harness the power of machine learning and computer vision for your applications.

Using Open CV to perform Face Detection:-

Import OpenCV.

Load the pre-trained Haar Cascade classifier for faces.

Load the image.

Convert the image to grayscale.

Use the detectMultiScale function to detect faces.

Draw rectangles around detected faces.

Display or save the result.

This simple process allows you to detect faces in images or video streams using OpenCV.

Performing Face detection on a still image:

1. Import OpenCV.

2. Load the image.

3. Initialize the pre-trained face detection model.

4. Convert the image to grayscale.

5. Detect faces using the model.

6. Draw rectangles around the detected faces.

7. Display or save the result.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.

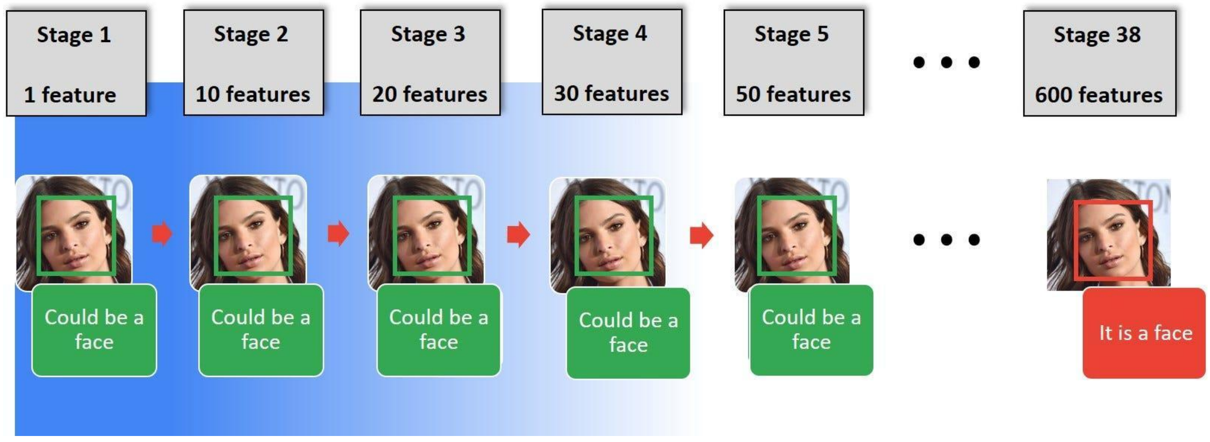


Table of contents

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.

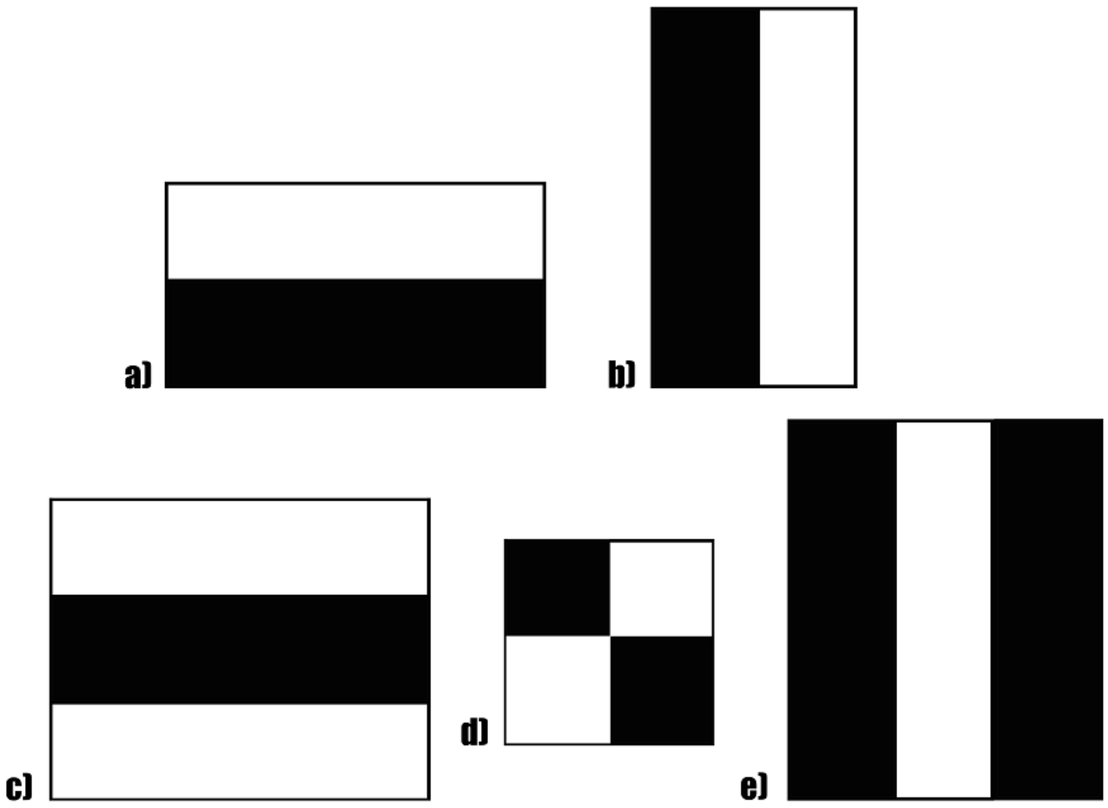
What is Haar Cascade Algorithm?

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.

Code:



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.

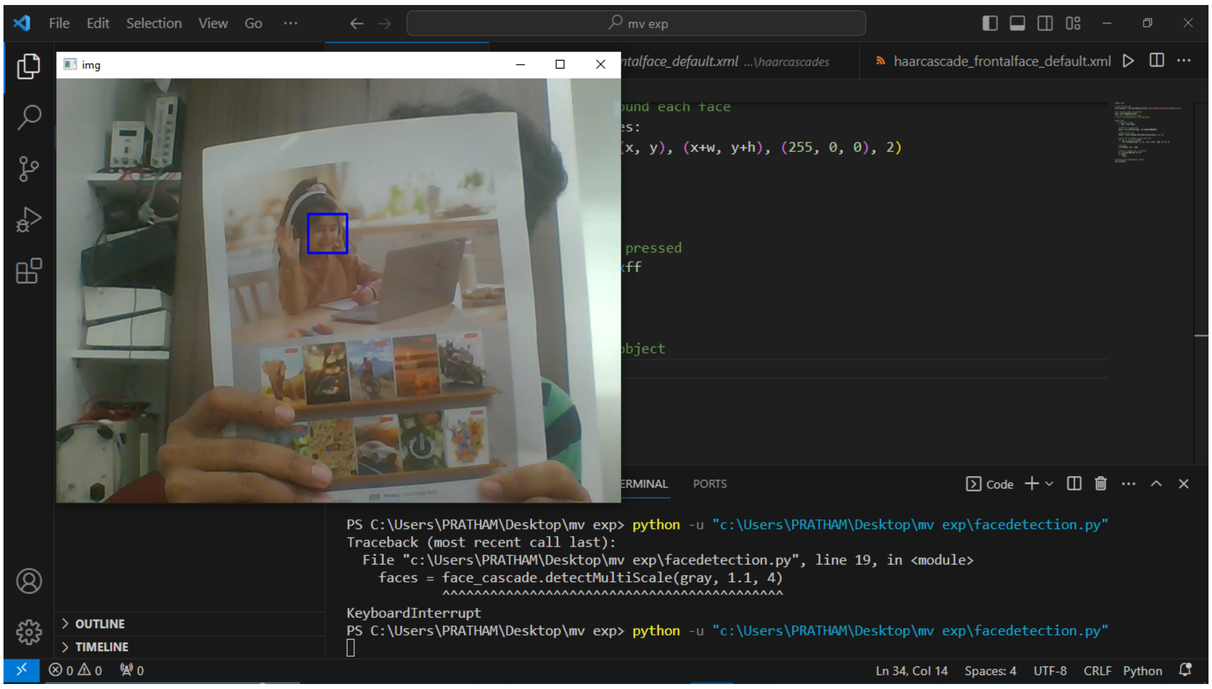
- Haar cascades are fast and can work well in real-time.
- Haar cascade is not as accurate as modern object detection techniques are.
- Haar cascade has a downside. It predicts many false positives.
- Simple to implement, less computing power required.

Code:

```
import cv2
```

```python
# Load the cascade

face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


# To capture video from webcam.

cap = cv2.VideoCapture(0)

# To use a video file as input

# cap = cv2.VideoCapture('filename.mp4')


while True:

    # Read the frame

    _, img = cap.read()


    # Convert to grayscale

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


    # Detect the faces

    faces = face_cascade.detectMultiScale(gray, 1.1, 4)


    # Draw the rectangle around each face

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

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


    # Display

    cv2.imshow('img', img)


    # Stop if escape key is pressed

    k = cv2.waitKey(30) & 0xff

    if k==27:
        break


# Release the VideoCapture object
cap.release()
```

Output:



File
Edit
Selection
View
Go
mv exp
img
ntalface_default.xml ...\haarcascades
haarcascade_frontalface_default.xml
ERMINAL
PORTS
Code
OUTLINE
TIMELINE

```
ound each face
es:
(x, y), (x+w, y+h), (255, 0, 0), 2)

 pressed
xff

bject
```

```
PS C:\Users\PRATHAM\Desktop\mv exp> python -u "c:\Users\PRATHAM\Desktop\mv exp\facedetection.py"
Traceback (most recent call last):
  File "c:\Users\PRATHAM\Desktop\mv exp\facedetection.py", line 19, in <module>
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

KeyboardInterrupt
PS C:\Users\PRATHAM\Desktop\mv exp> python -u "c:\Users\PRATHAM\Desktop\mv exp\facedetection.py"
```

Conclusion :-

Face detection on a still image is the process of using computer vision techniques, with OpenCV being a popular choice, to locate and identify human faces within a static image. This involves loading the image, initializing a pre-trained face detection model, converting the image to grayscale for better results, detecting faces in the image, and then drawing rectangles around the detected faces to highlight them. The final step is either displaying the image with the marked faces or saving it, depending on the intended use. This technology has numerous applications, including security systems, photography, and facial recognition.