



Vidyavardhini's College of Engineering &
technology

Department of Computer Engineering

| |
|--|
| Experiment No.6 |
| To study Detecting and Recognizing Faces |
| Date of Performance:21/08/2023 |
| Date of Submission: 04/09/2023 |

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 used for object detection in images. They are particularly valuable for real-time applications due to their speed and efficiency. The concept involves breaking down the detection process into stages. Each stage consists of a simple binary classifier trained to recognize specific image features, such as edges or textures, using Haar-like features.

During detection, a sliding window moves over the image, and at each stage, a classifier assesses whether the region within the window matches the object's features. If a region doesn't match, it's quickly rejected, reducing computational load. This "cascade" effect allows for fast rejection of non-object regions.

Haar Cascades are adaptable, allowing you to fine-tune parameters for accuracy and speed trade-offs. They are commonly used in face detection, but their flexibility makes them suitable for various object recognition tasks.

Getting Haar Cascade Data-

1. To obtain Haar Cascade data for object detection, follow these steps:
2. **Choose or Create a Positive Dataset:**Collect a set of images containing the object you want to detect.Create a text file listing the file paths of these positive images along with their corresponding object annotations (bounding boxes).
3. **Choose or Create a Negative Dataset:**Collect a set of images that do not contain the object of interest.Create a text file listing the file paths of these negative images.
4. **Install OpenCV:**Install OpenCV, an open-source computer vision library, which includes tools for creating Haar Cascades.
5. **Prepare Positive and Negative Samples:**Use OpenCV's tools to create positive samples and negative samples from your image datasets.Generate positive samples with annotations using the `opencv_createsamples` command-line tool.Create a text file listing the positive samples and negative samples.

6. **Train the Cascade Classifier:** Use the `opencv_traincascade` command-line tool to train the Haar Cascade classifier. Specify parameters like the number of stages, the size of positive and negative samples, and other training options. The tool will run through multiple iterations to improve the classifier's accuracy.
7. **Evaluate and Test the Cascade Classifier:** After training, evaluate the performance of your Haar Cascade classifier on test images. Adjust parameters and retrain if necessary to achieve the desired detection accuracy and false positive rate.
8. **Use the Trained Cascade Classifier:** Once satisfied with the performance, you can use the trained Haar Cascade XML file for object detection in your applications. OpenCV provides functions to load and apply the trained cascade classifier to detect objects in images or video streams.

Using Open CV to perform Face Detection:

To perform face detection using OpenCV, first, load a pre-trained Haar Cascade classifier for faces using `cv2.CascadeClassifier()`. Then, read an image or capture frames from a camera feed. Use the `detectMultiScale()` method to locate faces within the image or frame. Adjust the `scaleFactor` and `minNeighbors` parameters to control detection sensitivity and accuracy. Draw rectangles around the detected faces using `cv2.rectangle()`. Finally, display or save the processed image/frame to visualize or analyze the detected faces. This process allows for quick and efficient face detection in images or real-time video using OpenCV's computer vision capabilities.

Performing Face detection on a still image:

Performing face detection on a still image involves using computer vision techniques to identify and locate human faces within the image. Here's how it's done in a nutshell:

1. **Image Input:** Begin by loading the still image you want to analyze.
2. **Haar Cascade or Deep Learning Model:** You can choose between classic Haar Cascade classifiers or more modern deep learning models like Convolutional Neural Networks (CNNs). Haar Cascades work by sliding a window over the image and applying a series of classifiers to detect facial features. Deep learning models, like the popular OpenCV DNN module with pre-trained models, provide accurate face detection without the need for handcrafted features.
3. **Detection:** Apply the chosen model to the image. Haar Cascades will return bounding boxes around detected faces, while deep learning models may also provide facial landmarks for further analysis.

4. **Visualize and Post-process:** Draw bounding boxes or overlay facial landmarks on the image to highlight the detected faces. You can also perform additional tasks like gender or age estimation if required.
5. **Output:** The result is an image with annotated faces or a list of coordinates representing the location of detected faces in 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.

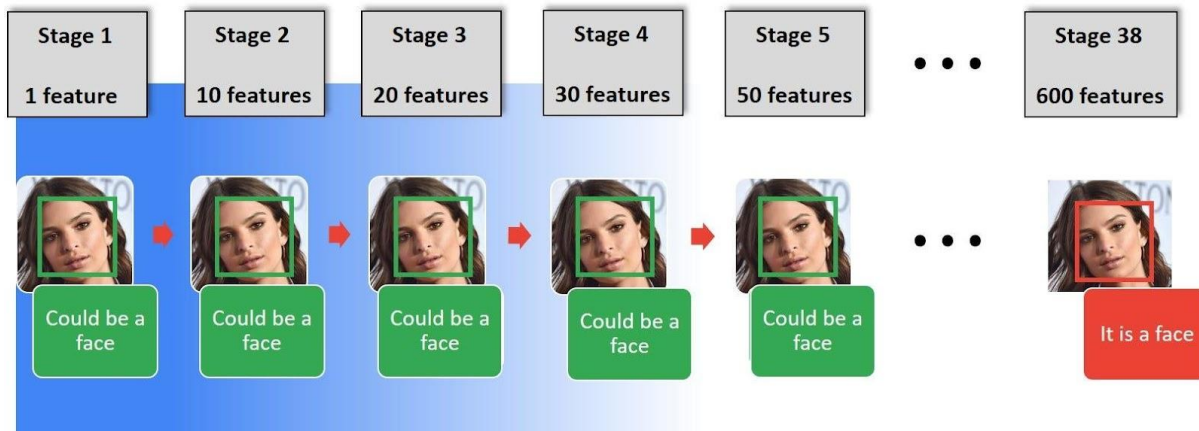


Table of contents

Why Use Haar Cascade Algorithm for Object Detection?

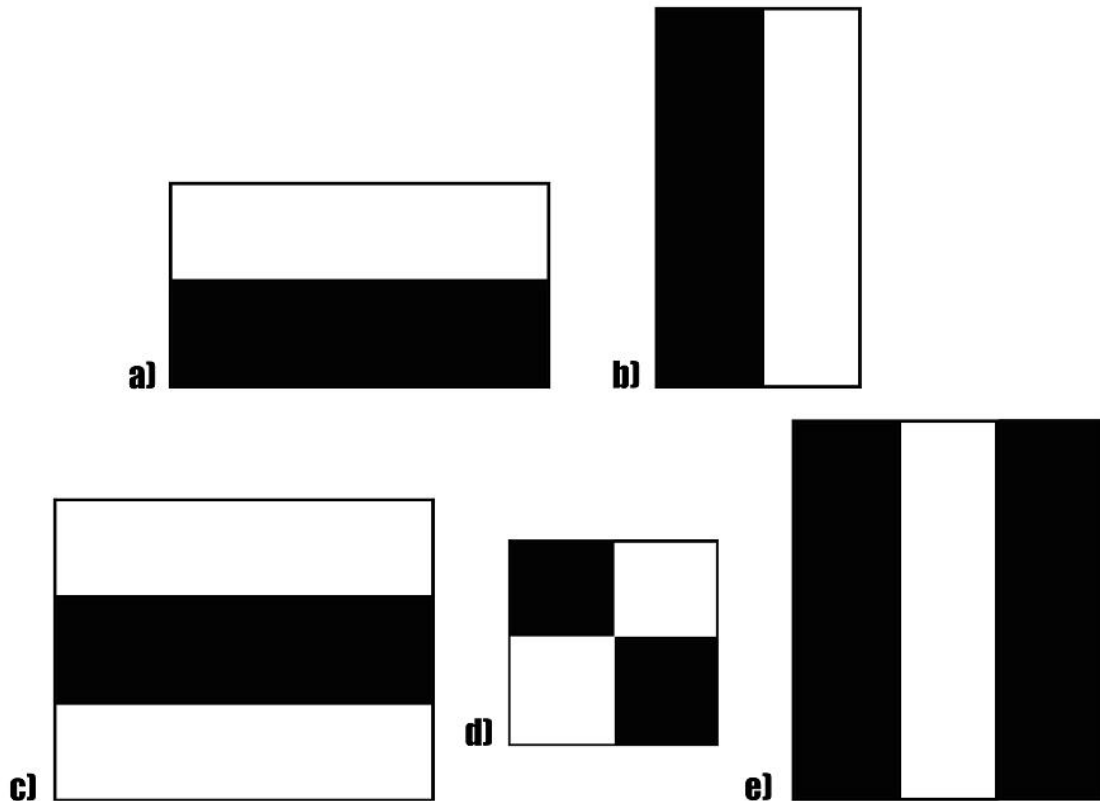
Identifying a custom object in an image is known as object detection. This task can be done using several techniques, but we will use the haar cascade, the simplest method to perform object detection in this article.

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:

```
pip install opencv-python
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.8.0.76)
```

```
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1.23.5)
```

```
import cv2
```

```
imagePath = 'input_image.jpg'
```

```
img = cv2.imread('mv.jpg')
```

```
img.shape
```

```
(225, 225, 3)
```

```
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
gray_image.shape
```

```
(225, 225)
```

```
face_classifier = cv2.CascadeClassifier(
```

```
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
```

```
)
```

```
face = face_classifier.detectMultiScale(
```

```
    gray_image, scaleFactor=1.1, minNeighbors=5, minSize=(40, 40)
```

```
)
```

```
for (x, y, w, h) in face:
```

```
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 4)
```

```
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(20,10))
```

```
plt.imshow(img_rgb)
```

```
plt.axis('off')
```

```
(-0.5, 224.5, 224.5, -0.5)
```

Output:



Conclusion :

In conclusion, the study of detecting and recognizing faces is a critical field within computer vision and artificial intelligence. It involves developing algorithms and techniques to locate and identify human faces within images or video streams. This technology finds wide-ranging applications, from security and surveillance to biometric authentication and entertainment. Achieving robust and accurate face detection and recognition remains an ongoing challenge due to varying lighting conditions, poses, and occlusions. Nonetheless, continuous research and advancements in deep learning and Haar Cascades have significantly improved the reliability and performance of these systems, making them increasingly valuable in today's digital world.