T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLİŞİM TEKNOLOJİLERİ MÜHENDİSLİĞİ ANABİLİM DALI

PYTHON FOR ENGINEERING APPLICATION

FACE PORTION DETECTION IMAGE PROCESSING USING PYTHON

MD AL AMIN HOSSAIN

218264001009

JANUARY-2023

KONYA

Contents

1. INTRODUCTION

It has become increasingly critical and challenging in recent years to manage the security of information or systems. We are aware of several computer crimes, hacker offenses, and commercial security breaches in recent years. In all of these issues, criminals were taking advantage of these systems to get access control by hacking information such as ID cards, keys, passwords, and PIN codes. Now that these issues have a technological remedy, "real" separation is confirmed [1]. The term "bio-metrics" also applies to this technique. Entrance controls using biometrics are automatic systems for confirming or identifying, such as thumbprints, facemask structures, or certain characteristics of a person's performance, such his or her writing style or designs[1].

In vision-related applications, such as face identification, features monitoring, facial emotion recognition, face synthesis, head pose estimation, etc., facial features detection is a crucial stage. It aids in the development of social communication abilities [2]. Typically, facial characteristics include prominent areas that are simple to identify, such as the corners of the eyes, nostrils, and lips. The majority of programs for tracking facial expressions still assign initial feature points manually[2].

One biometric procedure that benefits from both high accuracy and low insensitivity is face recognition. Recent years have seen a rapid increase in the need for face matching and its related phases, such as detection. After this procedure, it has numerous anticipated applications in computer vision systems and programmed access control systems. Particularly, identifying the face, eyes, nose, and mouth is the initial stage in automatically matching faces. But due to numerous variances in image elements, such as different face poses, recognizing the face, eyes, nose, and mouth is not easily accomplished (front, non-front)[1].

In this experiment, the method for identifying faces, eye pairs, mouths, and noses was based on a two-step process. First, the image is filtered or examined to see if there are any faces or face parts in it[3]. If this is the case, the image is next reviewed to see if there are any faces or face parts, and if so, where they are located (or where they are). Identify and locate faces of people in images regardless of their point, scale, in-plane variation, angle, or position (out-of-plane variation)[2].

We demonstrated in this study how to use the python haar cascade object detector to create a frontal view face detection algorithm based on the Viola-Jones approach. utilizing the object vision system in python. A face detector called

CascadeObjectDetector was created and is set up to follow the input file's selected user organization model. With the aid of vision.CascadeObjectDetector function, the file is created .A collection of positive samples (windows with faces) and a set of negative pictures are used in the attentional cascade training [4]. The number of cascade layers and the function parameters were modified to produce a more accurate detector. Finally, the face detector's performances for various tuning parameters were examined.

This is how the paper was put together. Describe the purpose of this strategy in Section 2. Tools for this model's requirements were described in Section 3. The application of the concept is demonstrated in Section 4 using a Python cascade object detector. Section 5 displays the tested experimental output results together with a few data images.The conclusion is described in Section 6.

2. GOALS AND OBJECTİVES

Face Detection is the first and essential step for face recognition, and it is used to detect faces in the images. It is a part of object detection and can use in many areas such as security, bio-metrics, law enforcement, entertainment, personal safety, etc[5].It is a method of biometric identification that uses that body measures, in this case, face and head, to verify the identity of a person through its facial biometric pattern and data [6]**.** The technology collects a set of unique biometric data of each person associated with their face and facial expression to identify, verify and/or authenticate a person. There are so many goals and objectives of face portion detection approach[6].Such as,

- It is used to detect faces in real time for surveillance and tracking of person or objects. It is widely used in cameras to identify multiple appearances in the frame Ex- Mobile cameras and DSLR’s. Facebook is also using face detection algorithm to detect faces in the images and recognise them.
- face detection is also needed for facial recognition algorithms to recognize which parts of a picture should be used to render face prints and which parts should be ignored.
- Face detection is a very important technology that saves us from all these bothers or troubles because it lets facial identification be fully automated hence enhancing productivity while also raising the rate of accuracy.
- Some popular applications like Snapchat or Instagram allow their users to modify their faces with fun filters in real-time. This is made feasible by facial recognition

algorithms, which inform the apps that there is a face that may be tracked and modified on the screen.

- Face detection can also help to provide data such as tracking the number of customers who come to the store or are in the store at the same time. In this way, creating marketing strategies gets easier. In addition to this, it allows following the number of customers inside even in the cases of infection, such as the limit of the number of people in closed areas during the Covid-19 pandemic.
- It is possible to witness face detection in our daily lives since it is the basis of a large number of facial apps. Face recognition is used to unlock our smartphones, and this would not be practicable without face detection. Similarly, face recognition surveillance technologies, face blurring, picture tagging, photo filters, and many more also fall under it.

3. REQUİREMENTS TOOLS

For implementing the face portion detection image processing method, we need two types of apparatus. One is a hardware-related apparatus, and the second is a software-related apparatus. such as,

- Hardware Apparatus:
  - A computer with windows version 7 or up
  - A good CPU or precessor
  - İnternet connection
- Software Apparatus:
  - A software development environment like Anaconda, Pycharm, visual studio, Matlab etc.
  - Python with opencv, Matplotlib, Numpy library
  - An image in local device
  - Viola-Jones algorithm
  - HAAR Casecading classifier files for face, eye,nose, and mouth

I. OpenCV

Open source Computer Vision is basically a library, used for real-time Computer Vision. It supports a wide range of programming languages like Python, C, C++, Java etc and also supports different platforms including Windows, Linux, MacOS. By using it, one can process images and videos to identify objects, faces or even handwriting of a human[7]. When it integrated with various libraries, such as

Numpy, Python is capable of processing the OpenCV array structure for analysis. To identify image pattern and its various features we use vector space and perform mathematical operations on these features[7].

II. Haar Casecade Algorithm

It is a machine learning algorithm used to detect objects in a static image or a real time video based on the edge and line detection features proposed by Paul Viola and Michael Jones in 2001. The algorithm is given a lot of positive images consisting of faces and a lot of negative images not consisting of any face to train the model[7]. This pre-trained model is available at the OpenCV GitHub repository. The repository has some XML files where this model is stored. This includes models for face detection, eye detection, upper and lower body detection etc.The algorithm contains 4 stages: Haar Feature Selection, Creating Internal Images, Adaboost Training, Cascading Classifiers.

Haar wavelet is a sequence of rescaled square-shaped functions and it is very similar to Fourier-analysis. This was first proposed by Alfred Haar in 1909. These features are very similar to convolutional kernels. Haar features are the relevant features for face, eye,nose and mouth detection[8]. OpenCV's algorithm is currently using the following Haar-like features which are the input to the basic classifiers:

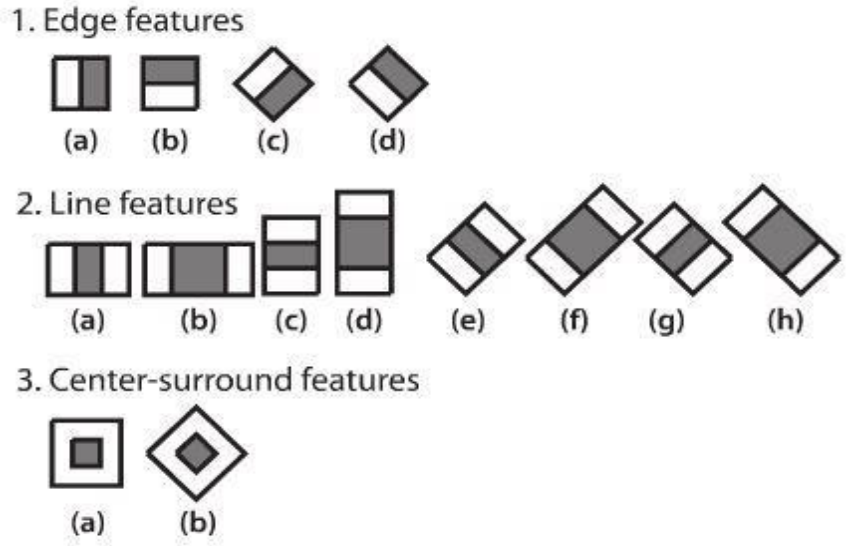

Figure 3.1: Haar feature basic classifiers

4. IMPLEMENTATION USING PYTHON

4.1. How To Start The Approach

There are many techniques to detect faces, with the help of these techniques, we can identify faces with higher accuracy. These techniques have an almost same procedure for Face Detection such as OpenCV, Neural Networks, Matlab, etc. The face detection work as to detect multiple faces in an image[9]. Here we work on OpenCV for Face Detection, and there are some steps that how face detection approach we can start, which are as follows-

- By specifying the picture's location, the image is first imported. The image is then converted from RGB to grayscale since grayscale makes it simple to identify faces but it not necessary.
- Then, if necessary, the photographs are resized, cropped, blurred, and sharpened by image manipulation.
- The next stage is image segmentation, which separates the various things present in a single image or is used for contour detection so that the classifier can swiftly identify the objects and faces in the image.
- The haar like features method, suggested by Voila and Jones for face detection, is the next stage. This method is used to locate the faces of people in a frame or picture. All human faces exhibit certain common characteristics, such as the nose area is brighter than the eye region and the eyes being darker than their neighboring pixels.
- The haar like algorithm is also used for selection extraction for an object in an image. Thanks to detection technologies, identifying eyes, noses, mouths, and other facial features of a face in the image are found. Because it is utilized to pick out the most important characteristics in a picture and extract them for facial detection.
- The face's position, or the area of interest in the image, is indicated in the following phase by providing the coordinates of x, y, w, and h. The region of interest where the face is detected can then be marked with a rectangular box.

4.2. Code Line Description

In this section we explained the code line of face portion detection for face, eye, nose, and mouth. First, we have to install OpenCV library which can be easily installed

by using the pip command*: **pip install opencv-python*** and then we will install the NumPy library by using the pip command: ***pip install numpy***.

To get started first we have to import the **NumPy** library named as **np** and **OpenCV** library named as **cv2**.

```
# import libraries
import cv2
import numpy as np
```

Now we will read the image in RGB format by using **imread** function and use **imshow** function to show the original image before detecting face,nose, mouth, and eyes in it as an output.

```
#read the image
image = cv2.imread('lena.jpeg')
# show the original read image
cv2.imshow("original image",image)
```

OpenCV library has so many pre-trained classifiers for face, eyes, smile etc. We can find these required XML files in Github. Now we will download those pre-trained face, eye and mouth detection model and save the files in the current directory. Now, include the ***haarcascade frontalface, haarcascade eye, haarcasecade nose, and haarcascade mouth*** XML files in the program.

```
#load the xml files for face, eye,nose and mouth detection
into the program
classifier_f =cv2.
CascadeClassifier('haarcascade_frontalface_default.xml')
classifier_n =cv2.
CascadeClassifier('haarcascade_mcs_nose.xml')
classifier_e =cv2. CascadeClassifier('haarcascade_eye.xml')
classifier_m =cv2.
CascadeClassifier('haarcascade_mcs_mouth.xml')
```

Now, we've to make the program identify face, nose, eye and mouth using haar based classifiers and we will use the **detectMultiScale** function where we'll pass **the input image** as a arguments.

```
#identify the face using haar-based classifiers
bboxes_f = classifier_f.detectMultiScale(image)
#identify the nose using haar-based classifiers
bboxes_n = classifier_n.detectMultiScale(image)
#identify the eye using haar-based classifiers
bboxes_e = classifier_e.detectMultiScale(image)
```

```
#identify the mouth using haar-based classifiers
bboxes_m = classifier_m.detectMultiScale(image)
```

Now, iterate through the faces array and draw a rectangle.In the code ***x2, y2 = x + width, y + height*** this line indicates that it basically selects row starting with y till y+height and column starting with x till x+width. We use these values to draw a rectangle using the built-in rectangle() function.

```
#iteration through the face array and draw a rectangle
for box in bboxes_f:
    x, y, width, height = box
    x2, y2 = x + width, y + height
    cv2.rectangle(image, (x, y), (x2, y2), (0,255,0), 2)
```

```
#iteration through the nose array and draw a rectangle
for box in bboxes_n:
    x, y, width, height = box
    x2, y2 = x + width, y + height
    cv2.rectangle(image, (x, y), (x2, y2), (0,255,0), 2)
```

```
#iteration through the eye array and draw a
rectangle
for box in bboxes_e:
    x, y, width, height = box
    x2, y2 = x + width, y + height
    cv2.rectangle(image, (x, y), (x2, y2), (0,0,255), 2)
```

```
#iteration through the mouth array and draw a rectangle
for box in bboxes_m:
    x, y, width, height = box
    x2, y2 = x + width, y + height
    cv2.rectangle(image, (x, y), (x2, y2), (255,0,0), 2)
```

Now, to show the final image after detecting face, eyes and mouth we will again use **imshow** function and will also give an infinite delay by using the **waitKey()** function.

```
#show the final image after detection
img=cv2.imshow('face,nose,eye and mouth detection image',
image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

5. EXPERIMENTAL OUTPUTS

We illustrated the experimental result outputs for face parts such as the eye, nose, and mouth in this section. We used the 281x309 RGB image size in this case.This study showed separate detection results for eyes, nose and mouth as well as full face detection with various parts.

Output for original image:

Figure 5.1: Original image

Output for face detection image:

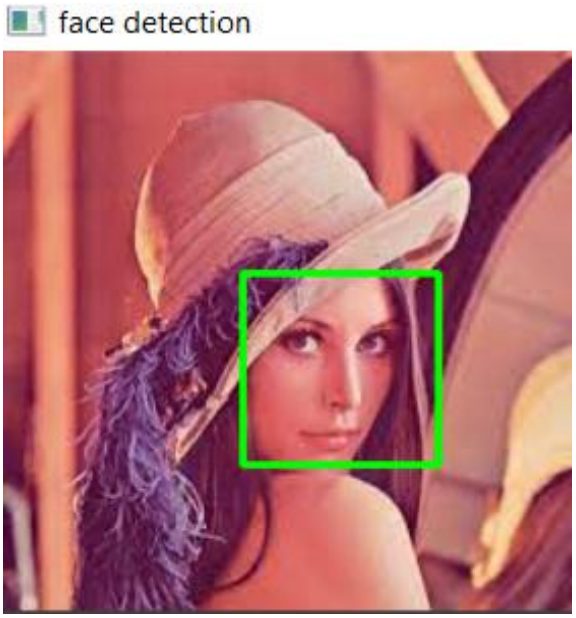

Figure 5.2: Face detection image

Output for eye detection image:

Figure 5.3: Eye detection image

Output for nose detection image:

Figure 5.3: Nose detection image

Output for mouth detection image:

Figure 5.3: Mouth detection imag

Output for facial parts detection image:

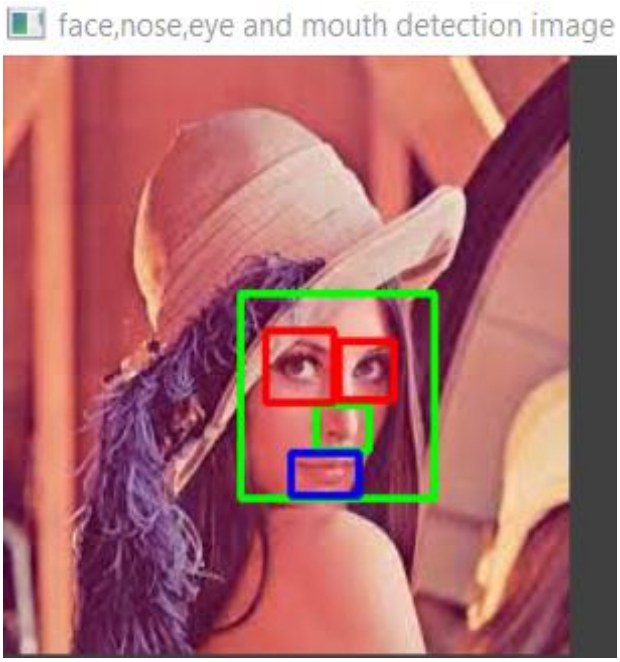

Figure 5.3: Face parts detection image

6. CONCLUDING REMARK

The method used in this study allows a human faces, eye pairs, noses, and mouths to be automatically identified in color photographs. This is based on a two-step process that first identifies areas in a color image where the human face, eyes, mouth, and nose are present. Information is then extracted from these areas, and the areas where the face, eyes, mouth, and nose are present are then identified. A color image with only the indicated image partsa face, a pair of eyes, a mouth, and a nose is completed. To eliminate item structures that would indicate the existence of an identified area, a combination of threshold holding, calculated values, and various functions are employed.

It is possible to witness face detection in our daily lives since it is the basis of a large number of facial apps. Face recognition is used to unlock our smartphones, and this would not be practicable without face detection. Similarly, face recognition surveillance technologies, face blurring, picture tagging, photo filters, and many more also fall under it. Using Face detection technology in these areas and more is also possible with Cameralyze. Cameralyze offers no-code user-oriented face detection applications with superior performance and high privacy.

7. REFERENCES

[1] Muhammad Nadeem Ali, "Face and Face Parts Detection in Image Processing," *Lahore Garrison Univ. Res. J. Comput. Sci. Inf. Technol.*, vol. 1, no. 1, pp. 62–68, 2017, doi: 10.54692/lgurjcsit.2017.01017.

[2] A. Majumder, L. Behera, and V. K. Subramanian, "Automatic and robust detection of facial features in frontal face images," *Proc. - 2011 UKSim 13th Int. Conf. Model. Simulation, UKSim 2011*, pp. 331–336, 2011, doi: 10.1109/UKSIM.2011.69.

[3] D. Lu and L. Yan, "Face Detection and Recognition Algorithm in Digital Image Based on Computer Vision Sensor," *J. Sensors*, vol. 2021, 2021, doi: 10.1155/2021/4796768.

[4] P. Griffin, "Understanding the Face Image Format Standards," *Identix Empowring Engineringering*, no. April, 2005.

[5] Maciej Serda *et al.*, "Synteza i aktywność biologiczna nowych analogów tiosemikarbazonowych chelatorów żelaza," *Uniw. śląski*, vol. 7, no. 1, pp. 343–354, 2013, doi: 10.2/JQUERY.MIN.JS.

[6] Alan Kilich, "5 Reasons Why Face Detection is Important?," *Cameralyze*, 2018. https://www.cameralyze.co/blog/5-reasons-why-face-detection-is-important (accessed Jan. 10, 2023).

[7] Parveen Kumar, "Face, eye and mouth detection with Python | by Parveen Kumar | Medium," *Medium*, Jul. 01, 2021. https://mittalparveen652.medium.com/face-eye-and-mouth-detection-with-python-458f2a028674 (accessed Jan. 10, 2023).

[8] Adrian Rosebrock, "OpenCV Haar Cascades - PyImageSearch," *Pyimagesearch*, Apr. 12, 2021. https://pyimagesearch.com/2021/04/12/opencv-haar-cascades/ (accessed Jan. 10, 2023).

[9] "OpenCV 3 Object Detection : Face Detection using Haar Cascade Classfiers - 2020." https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Object_Detection_Face_Detection_Haar_Cascade_Classifiers.php (accessed Jan. 10, 2023).