MINI PROJECT – I

(2018-19)

Multi Face Recognition

Mid-Term Report

Institute of Engineering & Technology

Team Members

Saurabh Chhonkar
(171500299)
Shubham Singh
(171500335)
Prabhat Kumar
(171500219)

Supervised By
Mr. Piyush vashistha
Asst. Professor
Department of Computer Engineering & Applications

Abstract

Identifying a person with an image has been popularised through the mass media. However, it is less robust to fingerprint or retina scanning. This report describes the multi face detection and recognition mini-project undertaken for the visual perception and autonomy module at Gla university. It reports the technologies available in the Open-Computer-Vision (OpenCV) library and methodology to implement them using Python. for face recognition Eigenfaces, Fisher faces and Local binary pattern histograms were used. The methodology is described including flow charts for each stage of the system. Next, the results are shown including plots and screen-shots followed by a discussion of encountered challenges. The report is concluded with the authors' opinion on the project and possible applications

1. Introduction:

The facial recognition has been a problem worked on around the world for many persons; this problem has emerged in multiple fields and sciences, especially in computer science, others fields that are very interested In this technology are: Mechatronic, Robotic, criminalistics, etc. platform .Net wrapper to the Intel OpenCV image processing library and C# .Net, library’s allow me capture and process image of a capture device in real time. The main goal of this article is show and explains the easiest way how implement a face detector and recognizer in real time for multiple persons using Principal Component Analysis(PCA) with eigenface for implementation it in multiple fields.

1.1 General Introduction to the topic:

The following document is a report on the mini project on Multi face recognition. It involved building a system for face detection and face recognition using several classifiers available in the open computer vision library(OpenCV). Face recognition is a non-invasive identification system and faster than other systems since multiple faces can be analysed at the same time. The difference between face detection and identification is, face detection is to identify a face from an image and locate the face. Face recognition is making the decision ”whose face is it ? ”, using an image database. In this project both are accomplished using different techniques and are described below.

1.2 Hardware and Software requirement:

Hardware requirement:

- Intel processor or any other processor
- 4GB Ram and 100GB disk space
- Web camera

Software requirement:

- Anaconda and spyder
- Python 3.7

2. Problem definition:

To verify multiple faces at a time from a image using Open Computer vision(Open CV).

3. Methodology:

Below are the methodology and descriptions of the applications used for data gathering, face detection, training and face recognition. The project was coded in Python using spyder and jupyter.

3.1 Face Recognition:

Facial recognition is a computer application composes for complex algorithms that use mathematical and matricial techniques, these get the image in raster mode(digital format) and then process and compare pixel by pixel using different methods for obtain a faster and reliable results, obviously these results depend of the machine use to process this due to the huge computational power that these algorithms, functions and routines requires, these are the most popular techniques used for solve this modern problem.

3.2 Face Detection:

Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Examples include upper torsos, pedestrians, and cars.
Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process.
A reliable face-detection approach based on the genetic algorithm and the eigen-face technique:
Firstly, the possible human eye regions are detected by testing all the valley regions in the gray-level image. Then the genetic algorithm is used to generate all the possible face regions which include the eyebrows, the iris, the nostril and the mouth corners.

3.3 Open Computer Vision(OpenCV):

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('watch.jpg',cv2.IMREAD_GRAYSCALE)
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

MINI PROJECT – I
(2018-19)

Multi Face Recognition

Mid-Term Report

Institute of Engineering & Technology

Team Members

Saurabh Chhonkar
(171500299)
Shubham Singh
(171500335)
Prabhat Kumar
(171500219)

Supervised By

Mr. Piyush vashistha
Asst. Professor
Department of Computer Engineering & Applications

Signature:_____________

4. Implementation:

- ### Training Data:

```
import cv2, os, sys
import numpy as np
import face_detect as face_detect

def training_data(data_folder):
    dirs = os.listdir(data_folder)
    faces = []
    labels = []
    for dir_name in dirs:
        if not dir_name.startswith("s"):
            continue;
        label = int(dir_name.replace("s", ""))
        subject_dir = data_folder + "/" + dir_name
        subject_images_names = os.listdir(subject_dir)
        for image_name in subject_images_names:
            if image_name.startswith("."):
                continue;
            image_path = subject_dir + "/" + image_name
            face, rect, length = face_detect.face_detect(image_path)
            if face is not None:
                faces.append(face[0])
                labels.append(label)

    return faces, labels
```

- ### Face Detect:

```
import cv2
import sys

def face_detect(imagePath):
# HaarCascade file, to detect the face.
faceCascade = cv2.CascadeClassifier("opencv-files/haarcascade_frontalface_alt.xml")
image = cv2.imread(imagePath)
#Convert image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
grays = []
```

```
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30)
    )

    # For drawing rectangles over multiple faces in the image
    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # Show Detected faces.
    cv2.imshow("Faces found", image)
    cv2.waitKey(1)

    # Append the detected faces into grays list.
    for i in range(0, len(faces)):
        (x, y, w, h) = faces[i]
        grays.append(gray[y:y+w, x:x+h])
    print("--------------------------------------------------------")
    print("Detecting Face -\-")
    return grays, faces, len(faces)
```

- **Face_recognition:**

```
import cv2

import numpy as np

import face_detect as face_detect

import training_data as training_data


label = []

def predict(test_img):

        img = cv2.imread(test_img).copy()

        print("\n\n\n")

        print("Face Prediction Running -\-")

        face, rect, length = face_detect.face_detect(test_img)

        print(len(face), "faces detected.")

        for i in range(0, len(face)):

                labeltemp, confidence = face_recognizer.predict(face[i])

                label.append(labeltemp)
        return img, label

faces, labels = training_data.training_data("training-data")
face_recognizer = cv2.face.LBPHFaceRecognizer_create()
face_recognizer.train(faces, np.array(labels))



# Read the test image.
test_img = "test-data/test.jpg"
predicted_img , label= predict(test_img)
cv2.destroyAllWindows()
cv2.waitKey(1)
cv2.destroyAllWindows()
print("Recognized faces = ", label)
```

.

5. PROGRESS TILL DATE & REMAINING WORK:

Some work on the code is remaining till now it is working but not perfectly to the output.

6. REFERENCES:

- www.wikipedia.com
- Machine learning and opencv
- www.google.com
- GeeksforGeeks.com