Faculty of Engineering

Credit Hours Engineering Programs

Mechatronics Engineering and Automation

Academic Year 2019/2020 – Spring 2019

CSE 489

Machine Vision

Project No. (2)

Face Detection and Tracking

Date due 16-Apr-19
Date handed in 16-Apr-19

Submitted by:

Ahmed **----------**

Contents

List of figures

1 ABSTRACT

This project is about detecting and tracking multiple faces using openCv library and python. The purpose of this code to detect on real time the faces and the facial landmarks of the person in front of the camera. An important library is used to help in facial detection is the dlib library which contains machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments.

2 TECHNICAL DISCUUSION

Every face has numerous, distinguishable landmarks, the different peaks and valleys that make up facial features. FaceIt defines these landmarks as nodal points. Each human face has approximately 80 nodal points. Some of these measured by the software are: Distance between the eyes Width of the nose Depth of the eye sockets The shape of the cheekbones The length of the jaw line These nodal points are measured creating a numerical code, called a faceprint, representing the face in the database.

Typically, 5 or 68 points are used for tracking a face. Even though these points are sufficient to track the approximate locations of facial landmarks, they are not sufficient to track the exact shape of facial landmarks

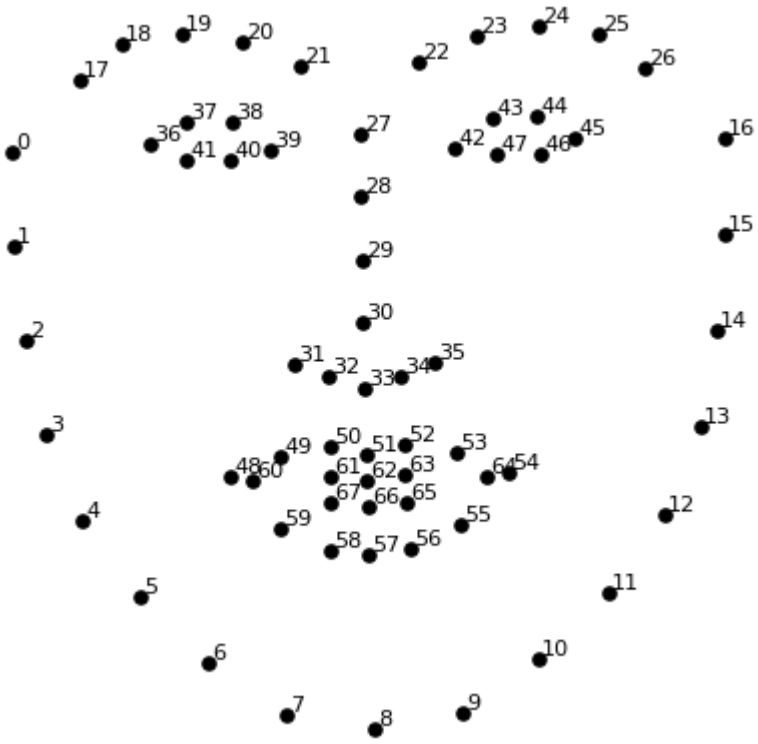

Figure 1 Facial Landmarks

Dlib is a general purpose cross-platform software library written in the programming language C++. Its design is heavily influenced by ideas from design

by contract and component-based software engineering. Thus it is, first and foremost, a set of independent software components. It is open-source software released under a Boost Software License.

The accuracy is calculated by counted the number of successfully detected faces in each frame then taking the average of the all accuracy divided by all frames.

3 RESULTS

Screenshots from the video stream showing different face poses.

Figure 2 and 3 show the face detection

Figure 2 Face Detection

Figure 3 Face detection and tracking

Figure 4 shows Facial landmark detection using Dlib library

Figure 5 shows the final output

Figure 4 Facial Landmarks detection and tracking

Figure 5 Final Output

The video link:

3.1 ACCURACY PER FRAME

```
Number of detected faces is/are :  1
accuracy is 0.9292929292929293
Real time Processing:  111.2255603
Number of detected faces is/are :  1
accuracy is 0.9293948126801153
Real time Processing:  111.3784882
Number of detected faces is/are :  1
accuracy is 0.9294964028776979
Real time Processing:  111.5321428
Number of detected faces is/are :  1
accuracy is 0.9295977011494253
Real time Processing:  111.6898406
Number of detected faces is/are :  1
accuracy is 0.9296987087517934
Real time Processing:  111.8491667
Number of detected faces is/are :  1
accuracy is 0.9297994269340975
Real time Processing:  112.0077279
Number of detected faces is/are :  1
accuracy is 0.9298998569384835
Real time Processing:  112.1613518
Number of detected faces is/are :  1
accuracy is 0.93
Real time Processing:  112.3165296
Number of detected faces is/are :  1
accuracy is 0.9300998573466477
Real time Processing:  112.4737027
final accuracy is 0.9300998573466477
[ WARN:0] terminating async callback
>>>
```

3.2 IS IT A REAL TIME PROCESS

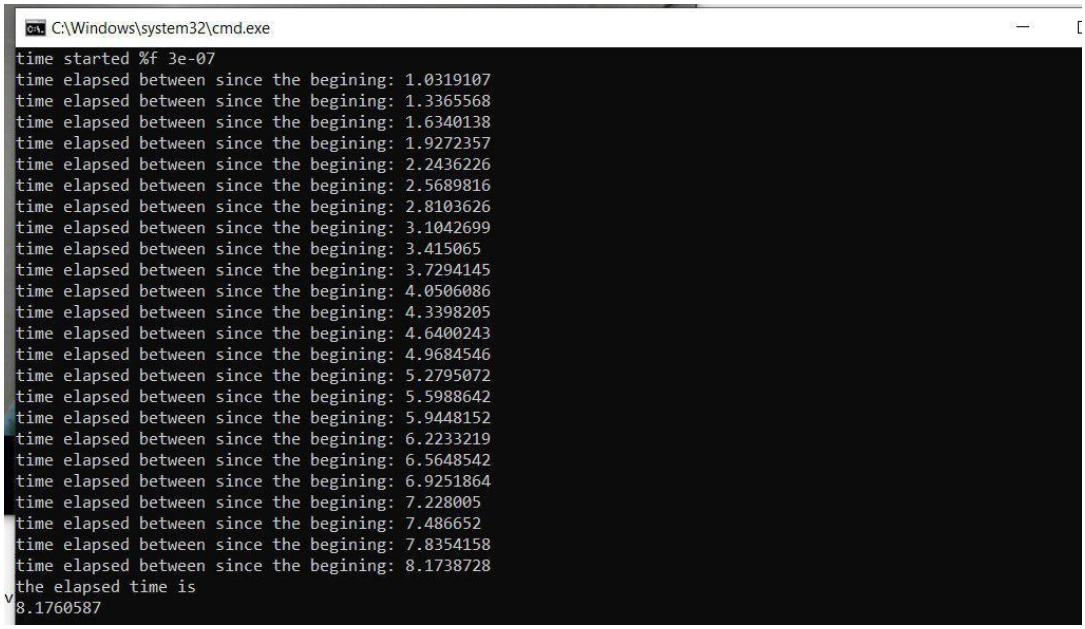

Figure 6 Real Time Code Execution

As we can see the difference between each iteration and the one after it does not exceed on average 300ms. So it a real time process for face detection

4 APPENDIX

```
import cv2

import dlib

import time

cap = cv2.VideoCapture(0)

detector = dlib.get_frontal_face_detector()

predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

e1=cv2.getTickCount()

#fourcc=cv2.VideoWriter_fourcc(*'XVID')

#out=cv2.VideoWriter('output.avi',fourcc,20.0,(640,480))

frameCount=0

numberOfDetectedFaces=0

acc=0

totalacc=0

GivenNumberOffaces=1

while True:

    ret, frame = cap.read()

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

  #  out.write(frame)

    time.sleep(0.1)
```

```
faces = detector(gray)
numberOfDetectedFaces=len(faces)
for face in faces:
    x1 = face.left()
    y1 = face.top()
    x2 = face.right()
    y2 = face.bottom()
    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 3)
    landmarks = predictor(gray, face)
    for n in range(0, 68):
        x = landmarks.part(n).x
        y = landmarks.part(n).y
        cv2.circle(frame, (x, y), 4, (255, 0, 0), -1)

frameCount=frameCount+1
cv2.imshow("Frame", frame)
e2=cv2.getTickCount()
t=(e2-e1)/cv2.getTickFrequency()
acc=numberOfDetectedFaces/GivenNumberOffaces
totalacc=totalacc+acc
print("Number of detected faces is/are : ",numberOfDetectedFaces)
```

```
    print("accuracy is" , totalacc/frameCount)
    print ("Real time Processing: ",t)
    key = cv2.waitKey(1)
    if key == 27:
        break

print("final accuracy is" , totalacc/frameCount)

#out.release()
cap.release()
cv2.destroyAllWindows()
```