OpenCV Face Detection

Marc Raymond A. Serrano, Computer Engineering

Mentor: Dr. Meng-Lai Yin

Kellogg Honors College Capstone Project

Abstract:

The purpose of this capstone project is to explore face detection. Face detection is a technology implemented in most camera-related applications which include autofocus, criminal detection in security cameras, and facial filters.
The face detection method studied is the “Haar cascade” which looks at an image with layers of predetermined shapes consisting of edges, lines, and centered shapes. If enough layers are verified then a box will be drawn around it to signify a face. In this project, the “Haar cascade” method is implemented on a Raspberry Pi which was set up to work with OpenCV 2.4.13, Python, and the Pi Camera.
The parts required to make a face detection device are available for purchase and requires programming knowledge to get the device working. Some applications of face detection include Snapchat filters, security camera criminal detectors, personal face recognition locks, and much more. However, the more complex the application, the stronger the hardware needed. This project accomplishes the base face detection required to make applications, such as the previously mentioned Snapchat filters and other face detection applications, which identifies faces and follows them.

Facial Recognition:

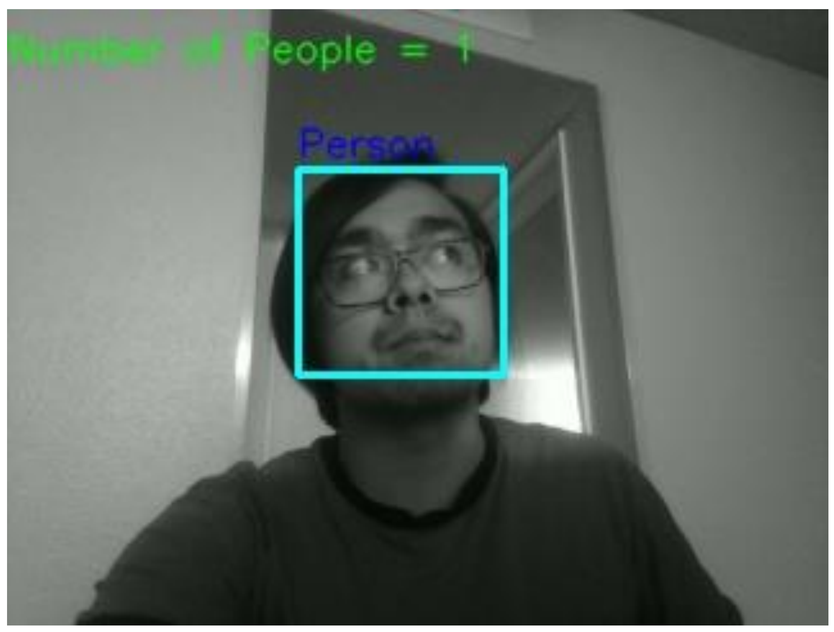
- Uses OpenCV Haar Cascade

1. Haar Cascade divides a picture into layers and decides if enough of those layers constitute a face
2. OpenCV already trained to detect a face
3. Ex. Eyes and eyebrows are darker in pictures so those are classified as a black rectangle while the area below that is usually lighter which is classified as a white rectangle.

Video Setup:

- Occurs at the top of the code

1. Calls all necessary libraries to use the Raspberry Pi camera
2. Sets color settings to be in grayscale
3. Sets resolution to 320x240
4. Sets video format to “bgr”
5. Settings 2 through 4 allow for better speed of video stream



Code Used in Project:

```
# Marc Raymond A. Serrano
# marc.serrano@ca.rr.com
# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import numpy
import io
import picamera

# allows conversion to numpy array
stream = io.BytesIO()

# haar cascade which detects faces
face_cascade = cv2.CascadeClassifier('/usr/share/opencv/haarcascades/haarcascade_frontalface_alt.xml')

# initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = (320, 240)
camera.color_effects = (128,128) #makes it grayscale in video
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(320,240))

#allows multiple pictures to be taken and saved without replacing the same picture each time
piccount = 0
# allow the camera to warmup
time.sleep(0.1)

# capture frames from the camera
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):

        # grab the raw NumPy array representing the image, then initialize the timestamp
        # and occupied/unoccupied text
        image = frame.array

        #Convert the picture into a numpy array
        buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)

        # convert to gray scale
        gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

        # finds faces
        faces = face_cascade.detectMultiScale(gray, 1.1, 5)

        # Draw a rectangle around every found face
        i = 0;
        for (x,y,w,h) in faces:
            i = i + 1;
            cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),2)
                        #(image, bottom left corner, top right corner, color settings, shift
            cv2.putText(image, "Person" ,(x,y-5),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,0,0),1,255)
            cv2.putText(image,"Number of People="+str(i),(0,20),cv2.FONT_HERSHEY_SIMPLEX,0.5,(0,255,0),1,255)
                        # show the frame
                        cv2.imshow( "Frame" , image)
                        key = cv2.waitKey(1) & 0xFF

                        # clear the stream in preparation for the next frame
                        rawCapture.truncate(0)
        # if the 'p' key was pressed, take a picture of the current screen
        # and save in current directory (/home/pi in my case)
        if key == ord( "p" ):
                piccount = piccount + 1
                cv2.imwrite( 'picture' + str(piccount) + '.jpg' ,image)
                        # if the `q` key was pressed, break from the loop
                        if key == ord( "q" ):
                                                break
```

Conclusion:

Ultimately, this project was a proof of concept that one can make a face detection device. This type of face detection is already used for practical applications such as surveillance and personal security. With additions to the code which I made, one can improve it to save one’s face to a computer and connect it to a motor-controlled lockbox to only open when the person’s face is seen by the Raspberry Pi camera.