

Beuth Hochschule für Technik Berlin

Detecting a face, eyes and smile through system camera and recording the footage

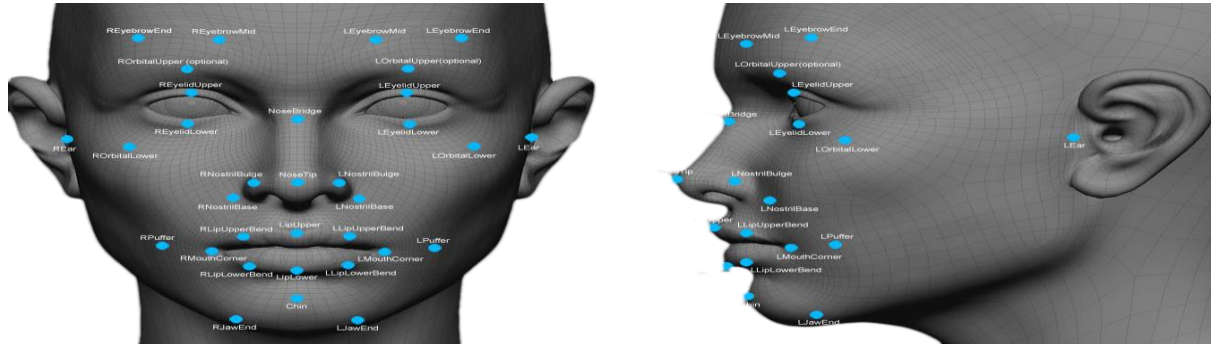
Faculty III – Civil Engineering – and Geoinformation
Course of Study Geoinformation

Submitted to: Lukasz Rojek

Submitted by: Prakash Neupane
 5th Semester
 Geodatendienste
 Matrikelnr. 863643
 Email: s70034@beuth-hochschule.de

Submitted on: 20 March 2020

Face detection is important in the field of Artificial Intelligence for identification, security and surveillance where eyes, nose, mouth of a human being are key features to detect a face... The distance between two eyes are almost same in each human being and the nose is symmetric to both side of the face, so these both features plays a significant role in face detection. After killing “**Osama Bin Laden**” by the US Army, facial recognition was used to identify his face/body.



Concept

A system camera that detects a face and the features like eyes and mouth for screening purposes or it can be used for snapping a photo for official purposes like for passport, detecting age, race, Identification card etc. Example for Biometric photos
Similarly this python program is able to recognize face and eyes from videos.

Preparing for the setup

- System Camera e.g. from Laptop Camera, external camera (Webcam), Drone Camera
- Python program, OpenCV, CascadeClassifier for face/eye/smile detection

Implementing codes with the system and camera

Initially HaarCascade (for eyes, mouth, nose etc.) depending upon the project, needs to be downloaded, installed and loaded, since HaarCascade is a classifier that helps in detecting object for which it has been trained through various algorithm pattern.

```
4
5 face_cascade = cv2.CascadeClassifier("C:/Users/Prakash/Desktop/Python/HaarCascade/haarcascade_frontalface_default.xml")
6 eye_cascade = cv2.CascadeClassifier("C:/Users/Prakash/Desktop/Python/video_2020/haarcascade_eye.xml")
7 #mouth_cascade = cv2.CascadeClassifier("C:/Users/Prakash/Desktop/Python/video_2020/haarcascade_mouth.xml")
8 smile_cascade = cv2.CascadeClassifier("C:/Users/Prakash/Desktop/Python/video_2020/haarcascade_smile.xml")
9
10
```

Fig. 2 HaarCascades for face, eyes, mouth and smiles

```
Vidcap = cv2.VideoCapture(0) #to capture from video ('sources filelink instead of 0')
```

Where “0” denotes the camera is being used from the system camera like that one from Laptop. If the Camera is to be used from the external one like “Webcam” then “1” has to be given.

Likewise instead of using the Camera, if a Video is to be used through which a face could be recognized then a “source link” has to be given instead of a value (0).

Here, the image captured from the Camera, which originally gray is converted to the color image, whereas Scalefactor and minNeighbours specify the reduced image scale and how many neighbors each candidate should have respectively.

```
while True:
    ret, img = Vidcap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor = 1.3, minNeighbors = 3)
```

Here, through Videowriter code, the video file is captured and saved as “File_name.avi” file in a working folder. In pic: “video4.avi”

```
fourcc = cv2.VideoWriter_fourcc(*'XVID') #defining the code to save the videowriter #newly added
output = cv2.VideoWriter('video4.avi',fourcc, 20.0, (640,480)) # newly added

check,frame = Vidcap.read()

while Vidcap.isOpened():          #: #instead of 1, here i have added Vidcap.isOpened
    ret, img = Vidcap.read()
    if ret ==True: #if pani added
        frame = cv2.flip(frame, 1) #Frame pani newly added 0 ko thau ma 1 lekhekeo
        output.write(frame) #added newly
```

Likewise, the “Region Of Interest” (ROI) is considered more than 70% while creating the “Bounding Box” so that the AI (Artificial Intelligence) could easily avoid mouth. Here, (ROI) is 80%. For all the tested value from 1.1-1.9, 1.8 was the best valued tracked in the program.

```
for (x,y, w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),3)

    roi_gray = gray[y:y-int((y+h)*0.8), x:x+w] #for avoiding mouth considering as eyes so region of interest more than 80%
    roi_color = img[y:y-int((y+h)*0.8), x:x+w]

    eyes = eye_cascade.detectMultiScale(roi_gray)
    smiles = smile_cascade.detectMultiScale(roi_color, 1.7, 20)

    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

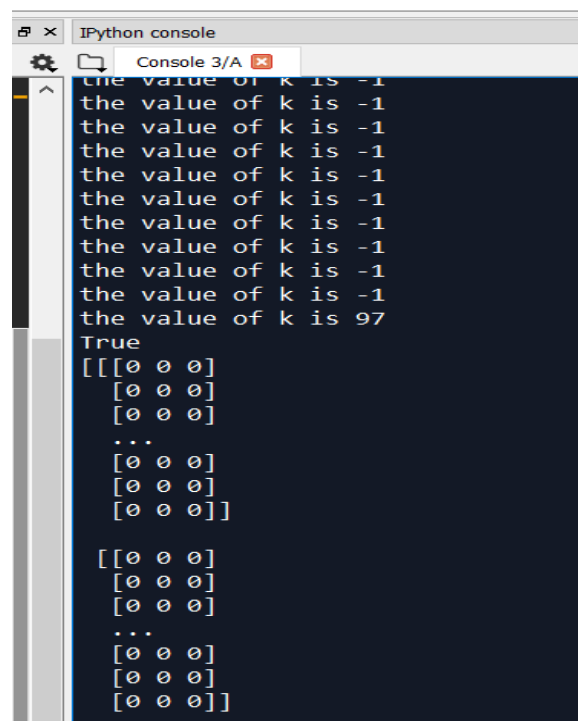
    for (sx, sy, sw, sh) in smiles:
        cv2.rectangle(roi_color,(sx, sy),(sx + sw), (sy + sh)), (0,0,255), 2)
```

Here with the k value, how a frame returns for specified no. of milliseconds (25). It either returns code of pressed key or -1. If the printed value $k > 0$, means a key was pressed. Through “cv2.imshow” the output is displayed. Whereas pressing “a” key allows the camera to be closed. Here the “Vidcap.release ()” allows the camera to be released so that it could be used for other purposes too. Since python was able to display image here, boolean character is “True”. Here time.sleep (3) helps in capturing the object image frame for 3 seconds. Numpy arrays are shown here in 2D, whereas frame is n –dimensional array.

```

37
38     k = cv2.waitKey(25) #how a frame for specified no. of milliseconds. returns either
39     print("the value of k is " + str(k)) ## check if the return value of waitkey() is a
40     if int(k) == 97: # if you press "a" then close.
41         break
42
43     cv2.imshow('img',img) #Displaying the output
44
45     print(check)
46     print(frame)
47     time.sleep(30)
48
49     Vidcap.release() #with no release function camera may not be free for another purposes
50     cv2.destroyAllWindows()
51     Vidcap.stop() ## TODO: stop the cam

```



```

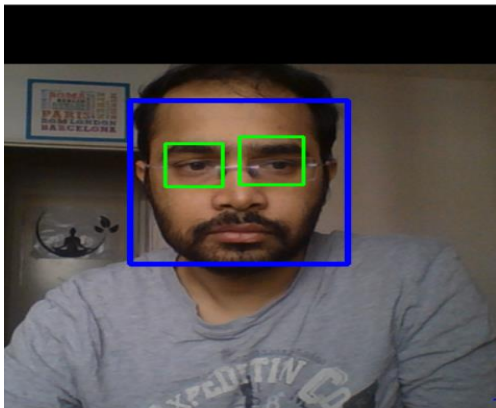
the value of k is -1
the value of k is -1
the value of k is -1
the value of k is -1
the value of k is -1
the value of k is -1
the value of k is -1
the value of k is -1
the value of k is -1
the value of k is 97
True
[[[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]]

[[[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]]

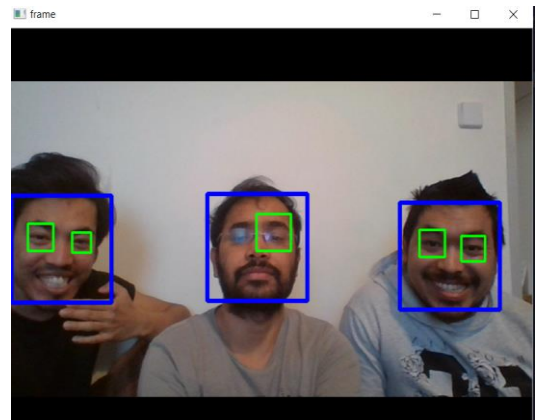
```

Variable explorer			
Name	Type	Size	Value
check	bool	1	True
eh	int32	1	int32 object of numpy module
ew	int32	1	int32 object of numpy module
ex	int32	1	int32 object of numpy module
ey	int32	1	int32 object of numpy module
eyes	int32	(2, 4)	ndarray object of numpy module
faces	int32	(1, 4)	ndarray object of numpy module
frame	uint8	(480, 640, 3)	[[[0 0 0] [0 0 0] [0 0 0] ... [0 0 0] [0 0 0] [0 0 0]]]
gray	uint8	(480, 640)	[[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] ... [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]]]
h	int32	1	int32 object of numpy module
img	uint8	(480, 640, 3)	[[[0 0 0] [0 0 0] [0 0 0] ... [0 0 0] [0 0 0] [0 0 0]]]

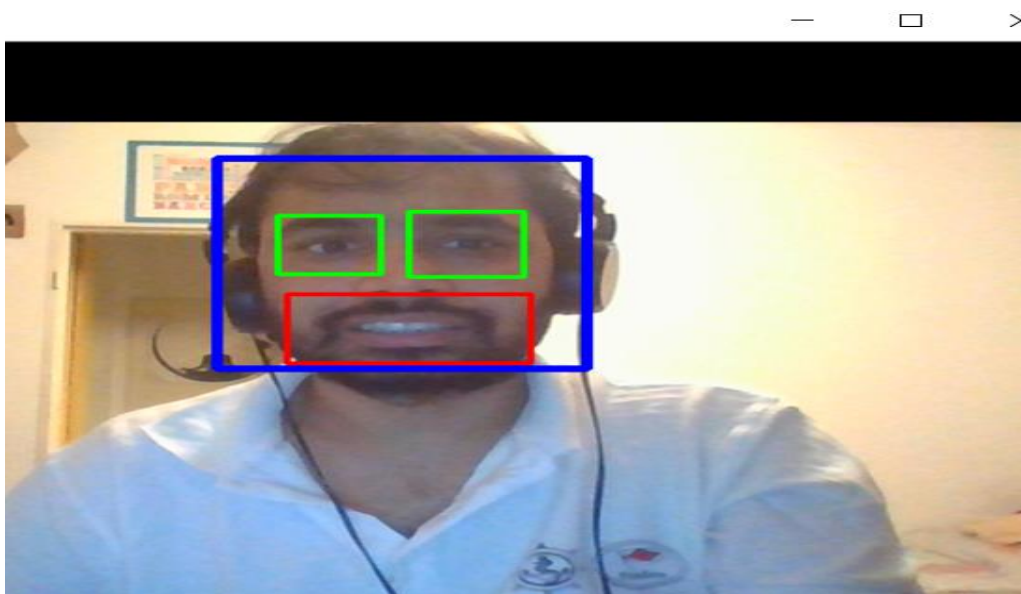
Here are some images after completing the project:



Face and eye detected when not smiling

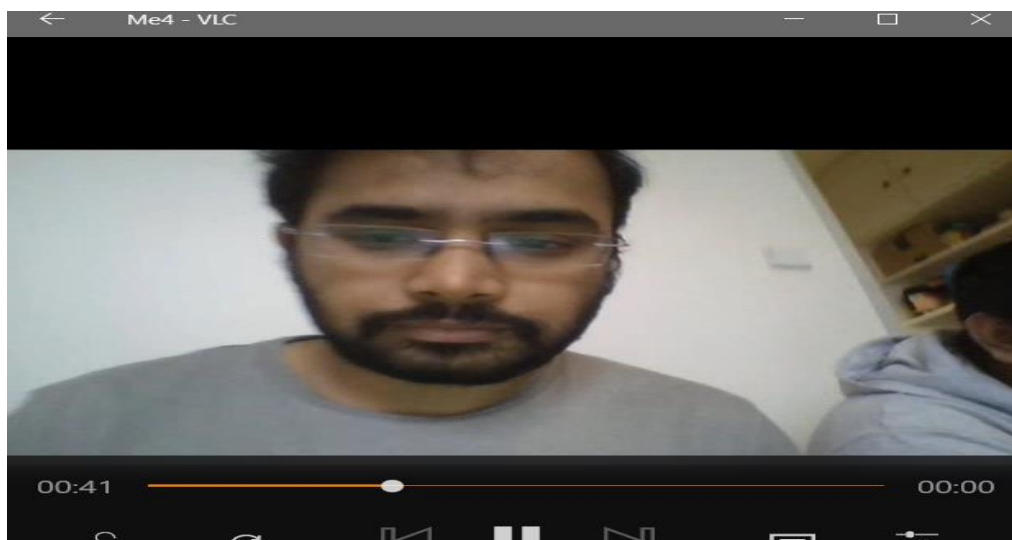


Detection in a group of people



Face, eyes and smile is detected

And finally a captured video files:



Sources:

Haarcascade: <https://github.com/opencv/opencv/tree/master/data/haarcascades>

Numpy : Numpy.org

Opencv : Opencv.org/ tutorials

<https://www.facefirst.com/blog/brief-history-of-face-recognition-software/>

stackoverflow.com

Images: <https://www.rhombussystems.com/blog/ai/rhombus-product-insight-%E2%80%93-face-detection-how-our-security-camera-system-captures-the-best-face/>

Complete Code:

```
import numpy as np
import cv2
import time
#import opencv-contrib

face_cascade =
cv2.CascadeClassifier("C:/Users/Prakash/Desktop/Python_/haarcascade_frontalface_default.xml")
eye_cascade =
cv2.CascadeClassifier("C:/Users/Prakash/Desktop/Python_/haarcascade_eye.xml")
#mouth_cascade =
cv2.CascadeClassifier("C:/Users/Prakash/Desktop/Python_/haarcascade_mouth.xml")
smile_cascade =
cv2.CascadeClassifier("C:/Users/Prakash/Desktop/Python_/haarcascade_smile.xml")

Vidcap = cv2.VideoCapture(0) #to capture from video ('sources filelink instead of 0')

fourcc = cv2.VideoWriter_fourcc(*'XVID') #defining the code to save the videowriter #newly
added
output = cv2.VideoWriter('here_filename.avi',fourcc, 20.0, (640,480)) # captured video
source, video width and height

check,frame = Vidcap.read()

while (Vidcap.isOpened()):          #: #instead of 1, here i have added Vidcap.isOpened-->
returns False
    ret, frame = Vidcap.read()
    if ret ==True:
        frame = cv2.flip(frame, 1) # if frame is 0, videoframe rotates to upside down
        output.write(frame)

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #to capture in monochrome
    faces = face_cascade.detectMultiScale(gray, scaleFactor = 1.5, minNeighbors = 5)

    for (x,y, w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),3)
```

```
roi_gray = gray[y:y-int((y+h)*0.7), x:x+w] #for avoiding mouth considering as eyes so
region of interest more than 80%
```

```
roi_color = frame[y:y-int((y+h)*0.7), x:x+w]
```

```
eyes = eye_cascade.detectMultiScale(roi_gray)
```

```
smiles = smile_cascade.detectMultiScale(roi_gray, 1.4, 20) #by testing all the value
from 1.1-1.9, 1.6 was the best one
```

```
for (ex,ey,ew,eh) in eyes: #bounding box for eye detection
```

```
cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
```

```
for (sx, sy, sw, sh) in smiles: #bounding box for smile detection
```

```
cv2.rectangle(roi_color,(sx, sy),((sx + sw), (sy + sh)), (0,0,255), 2)
```

```
k = cv2.waitKey(25) #how a frame for specified no. of milliseconds. returns either code of
pressed key or -1.
```

```
print("the value of k is " + str(k)) ## check if the return value of waitkey() is a number if
yes. if k>0 means some key was pressed.
```

```
if int(k) == 97: # if you press "a" then close.
```

```
break
```

```
#else: #newly added
```

```
#break #newly added
```

```
cv2.imshow('frame',frame) #Displaying the output
```

```
print(check)
```

```
print(frame)
```

```
time.sleep(3) # captures images for 3 seconds after pressing the stop button
```

```
#cv2.imshow('frame', frame) # newly added
```

```
output.release
```

```
Vidcap.release() #with no release function camera may not be free for another purposes
```

```
cv2.destroyAllWindows()
```

```
Vidcap.stop() ## TODO: stop the cam
```