Practical Assignment № 4

Face Detection using Viola-Jones Approach.

Instructing Professor :Sergei Shavetov & Andrei Zhdanov
Author :Xu Miao

June 1, 2022

Contents

1 Introduction

§1.1 Student information

- ★ Name: Xu Miao
- ★ ITMO Number: 293687
- ★ HDU Number: 19322103

§1.2 Purpose.

Study of Viola-Jones approach for detection of faces and part of bodies in the images.

§1.3 Introduction

In this report, I will complete the content of the following sections:

1) **Faces detection.** (3 images.)
   a) search faces using Viola-Jones approach
   b) Calculate the number of found faces on each image.
2) **Body parts detectiong.** (3 images.)
   a) search at least two parts of bodies in the one image (e.g. eyes, mouths, noses)
   b) increase the accuracy use ROI (upper part of bodies or faces)
   c) Calculate the found elements in each category.
3) **Optional 1.**
   a) Implement the face detection in videostream using pre-recorded video with faces.
4) **Optional 2.**
   a) Implement the face detection in live videostream using web-camera.

2 Notations

Table 2.1: Notation used in this report

Symbol	Definition
T	training set
$D_1(i)$	the weights for a training set
ϵ_k	error
α_k	the weight of the currently selected weak classifier
TP	true positive rate

★ Other notations instructions will be given in the text.

3 Solution of problem

§3.1 Faces detection.

The Viola-Jones face detector method is based on the following concepts:

1) Haar-like feaures as weak classifiers.
2) Integral image representation for fast calculation of Haar-like features.
3) AdaBoost training method to combine weak classifiers into a strong classifier.
4) Combining of strong classifiers into a cascade classifier.

§3.1.i Theorem 1 : Haar-like feature

Haar-like features

1) **Haar-like features** Haar-like feature is a kind of a weak classifier. It can be defined as the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image. In a traditional Viola-Jones face detector algorithm 4 types of Haar-like features that are shown in Fig.3.1 are used.

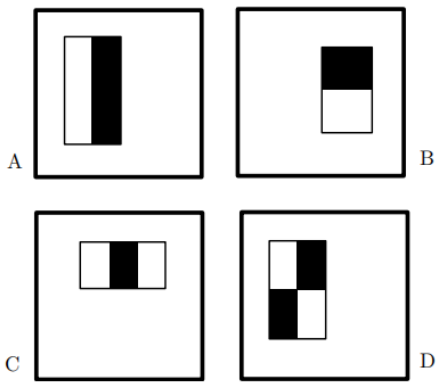

Fig. 3.1. Haar-like features used in Viola-Jones face detector

2) **A Quick Calculation Method: Integral Image**

To calculate the value of the Haar-like feature we need to calculate sums of pixels inside rectangular areas of the image and do it as fast as possible.

$$\text{value} = \sum(\text{ pixels in black area }) - \sum(\text{ pixels in white area }). \tag{3.1}$$

Obviously, the straightforward calculation of the sum of pixel values in a rectangle would require number of sums that is equal to number of pixels minus one. To speed up the feature calculation, an integral image representation is used. In this

representation each pixel stores the sum of all pixel values that are positioned to the left and above of the current pixel.

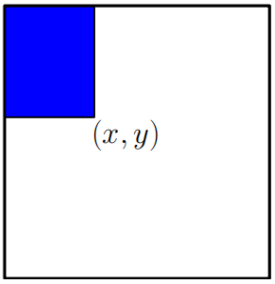

Fig. 3.2. Integral image

To calculate the sum of pixel intensity values in an arbitrary rectangle we need to access four pixels of an integral image which are located at the corners of the rectangle, see Fig.3.3.

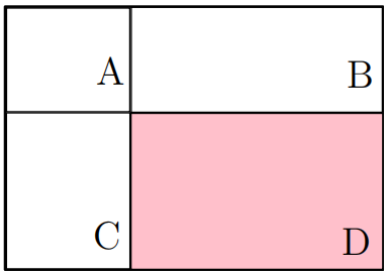

Fig. 3.3. Rectangular sum calculation with an integral image

where D is the bottom right corner of the rectangle, B is a pixel one pixel above the top right corner of the rectangle, C is the pixel to one pixel the left of the bottom left corner of the rectangle, and A is a pixel one pixel above and to the left of the top left corner of the rectangle.

3) **training algorithm : AdaBoost.**

The set of Haar-like features (which are weak classifiers) can be combined with a weighted sum of their values to form a more complex strong classifier. The training algorithm is called AdaBoost. It consists of several boosting rounds, and each boosting round is a selection of a best weak Haar-like feature to classify the training set with taking the classification errors of the previous rounds into an account.

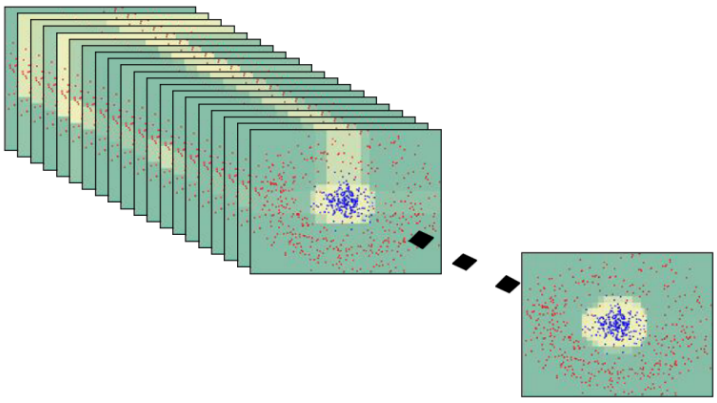

Fig. 3.4. Combining weak classifiers into a strong classifier

Formally, the AdaBoost training scheme algorithm can be described with following steps:

1) On an input we have a training set $T = \{(x_i, y_i) \mid x_i \in X, y_i \in \{-1, 1\}\}$ and a set of all possible weak classifiers $\{h\}$.
2) Initialize the weights for a training set items to be equal and sum up to 1.$D_1(i) = 1/m$, where m is a number of training set items.
3) Do K iterations:
   a) Choose h_k from a set of weak classifiers H, so that the weighted classification error probability is minimal (the probability of the wrong classification with taking weights into an account):

$$\epsilon_k = \Pr_{iD_k} [h_k(x_i) \neq y_i] \tag{3.2}$$

   b) Calculate the weight of the currently selected weak classifier basing on its classification error probability:

$$\alpha_k = \frac{1}{2} \ln \left(\frac{1 - \epsilon_k}{\epsilon_k} \right) \tag{3.3}$$

   c) Reweigh the training set with new weights:

$$D_{k+1}(i) = \frac{D_k(i)}{Z_k} \cdot \begin{cases} e^{-\alpha_k}, h_k(x_i) = y_i \\ e^{\alpha_k}, h_k(x_i) \neq y_i \end{cases} \tag{3.4}$$

4) After completing K iterations build a strong classifier as a weighted sum of weak classifiers that were selected during boosting rounds:

$$H(x) = \operatorname{sign} \left(\sum_{k=1}^{K} \alpha_k h_k(x) \right) \tag{3.5}$$

§3.1.ii Theorem 2: Cascade classifiers.

Cascade classifiers)
A strong classifier that have a required accuracy may require calculation of too much weak classifiers that would slow down the detection speed taking into an account that most of scanned windows do not contain faces. To speed up the detection rate a set of classifiers with increasing complexity are organized in a cascade of classifiers. Thecascade contains a set of classifiers with an increasing complexity and detection rate, see Fig.3.5.

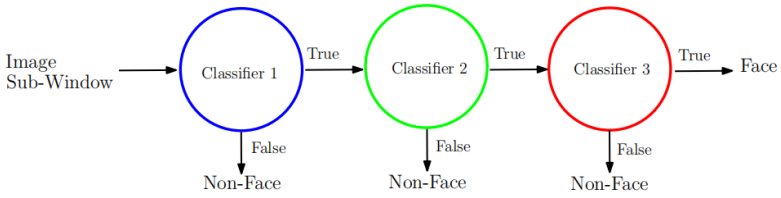

Fig. 3.5. Cascade classifier

To be classified positively, a sliding window should pass all cascade stages. In case if any classifier rejects the window, it is immediately rejected and detector proceeds to the next window. As a result, that most of negative windows are rejected fast with first fast classifiers in the cascade.
The detection rate (true positive rate, TP) of a cascade classifier is a multiplication of detection rates of all classifiers in a cascade:

$$TP = \prod_i TP_i \tag{3.6}$$

The false positive rate (FP) is also a multiplication of false positives of cascade classifiers;

$$FP = \prod_i FP_i \tag{3.7}$$

As a result, to build a classifier with 0.9 true positive rate and 10^{-6} false negative, each classifier in a cascade should meet the requirement of of 0.99 true positive and just 0.3 false positive.
Each classifier of the cascade is trained using the AdaBoost training scheme with requirement to maximize the true positive detection rate with keeping false positive within a given range. The training set is modified between the boosting rounds to increase the complexity of each cascade step.

§3.1.iii Solution

Face detection.

1.a Selection of the original image

In order to study the adaptability of the Viola-Jones method, images with **all frontal faces** (original image 1), images with **profile faces** and **different skin** tones (original image 2) are selected, and the face images **in the animation** (original image 3) are as follows:

Fig. 3.6. original image 1

Fig. 3.7. original image 2

Fig. 3.8. original image 3

1.b Image processing result

Fig. 3.9. Image result 1

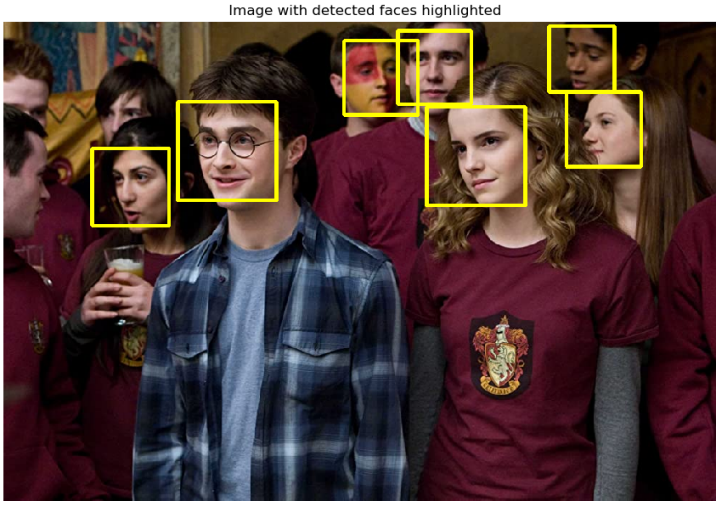

Fig. 3.10. Image result 2

Fig. 3.11. Image result 3

1.c full source code

You can click ⋆here⋆ to see the full code for this part.

1.d Comments

Table 3.1: face detection result

image	actual number of faces	number of detected faces
1	6	6
2	10	7
3	4	2

- ⋆ For standard frontal face images, the Viola-Jones approach performs well and detects faces accurately (even for infants) (Fig. 3.9)
- ⋆ The Viola-Jones approach can adapt to slight side faces (angle changes) and occlusion, but when the occlusion or the side face angle is large, it cannot accurately detect(Fig. 3.10)
- ⋆ In the face of face images in animation, the Viola-Jones approach can identify very standard face images (front view), and cannot achieve good results even for slight side faces (Fig3.11)

§3.2 Body parts detectiong.

§3.2.i Theorem.

Region-Of-Interest (ROI).
Sometimes in object detection, we don't need to scan the entire image.
For example, when we are detecting eyes,it's obvious that we don't need to scan the whole image for eyes as there would be a lot of false-popsitive results, so we will define a **Region-Of-Interest (ROI)** object as a face which was already found and then search for eyes with taking the **ROI** into an account.
When we are looking for eyes, nose, mouth, we can divide the face into three parts:

$$\begin{aligned} \text{face}_{\text{top}} &= I\left[y : y + \frac{3h}{2}, x : x + w\right] \\ \text{face}_{\text{bottom}} &= I\left[y + \frac{3h}{5} : y + h, x : x + w\right] \\ \text{face}_{\text{mid}} &= I\left[y + \frac{h}{4} : y + \frac{3h}{4}, x : x + w\right] \end{aligned} \tag{3.8}$$

We can perform object detection on the three corresponding ROIs.

§3.2.ii Solution

1.Body parts detectiong.

1.a Selection of the original image Use the same 3 images as the face detection part

1.b Image processing result

Fig. 3.12. Image result 1

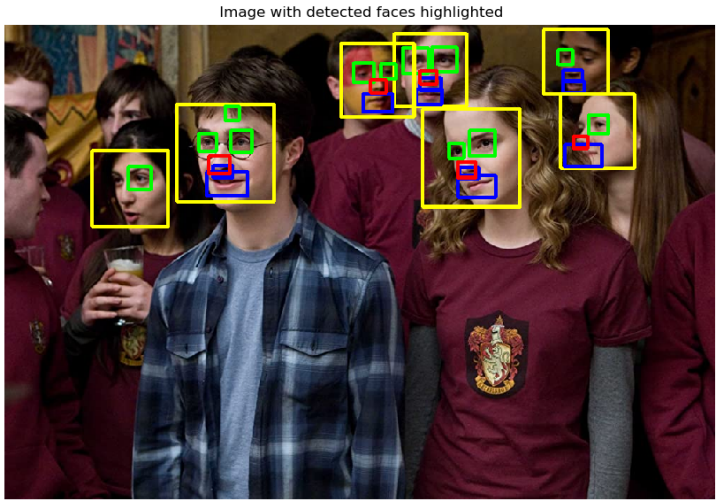

Fig. 3.13. Image result 2

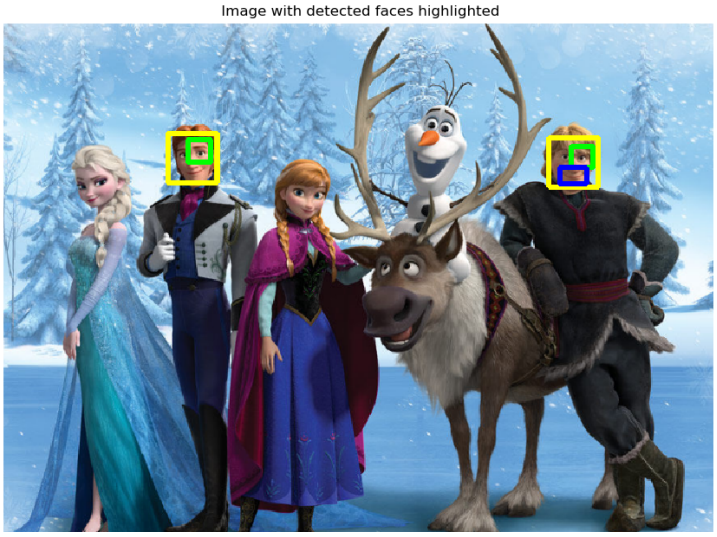

Fig. 3.14. Image result 3

1.c full source code

You can click ⋆here⋆ to see the full code for this part.

1.d Comments

Table 3.2: Face part detection result

image	actual number of eyes,noses,mouths	number of detected eyes,noses,mouths
1	12,6,6	12,6,6
2	16,9,8	13,5,9
3	8,4,4	1,0,1

- ⋆ From the results of the two experiments, whether it is face detection or face position detection, the Viola-Jones approach performs well for frontal face images, and is not affected by skin color, slight profile, and occlusion, but it cannot cope with large-area occlusions.(Fig3.11,3.12,3.13)
- ⋆ The advantage of using ROI is that false positive results can be avoided to a large extent. The disadvantage is that if the ROI is not detected, then the part above it has no chance to be detected.(Fig3.11,3.12,3.13)

§3.3 Optional.Face detection in video.

§3.3.i Theorem.

Face Detection in Video Streams.(pre-recorded video)
We all know that the video is composed of one frame by one frame, so the face detection based on the first part of the video stream is very simple and can be implemented by the following steps:

1. Read the video frame by frame
2. Perform face detection on each frame of image
3. Read and record the original video sound
4. Combine the processed images according to the frame rate of the original video and add sound and output to a new video file.

Face Detection in Video Streams.(using Web-camera)
Implementing face detection in a webcam is almost the same as in a recorded video, only the read and display positions are different.
I've added a fun little feature: replace faces with pictures of cats.Specifically, it can be achieved through the following steps:

1. Use $cv2.VideoCapture(0)$ to get the image of each frame of the camera
2. Face detection for each frame of image
3. Read in the cat image and scale it by the size of each face, then replace the pixels of each face.
4. display image
5. Loop 1-4 until the "q" key is pressed to exit

§3.3.ii Solution

1.face detection in videostream using pre-recorded video

1.a Selection of the original video

I edited a clip from my favorite family comedy "goodluck Charlie"

1.b Image processing result

I posted the processed video to a Chinese video site "bilibili" (similar to YouTube), you can click this link to view my video:

★bilibili video : Facial Recognition of Fragments from "Goodluck Charlie "(Viola-Jones Approach)★

Below are two screenshots from the video:

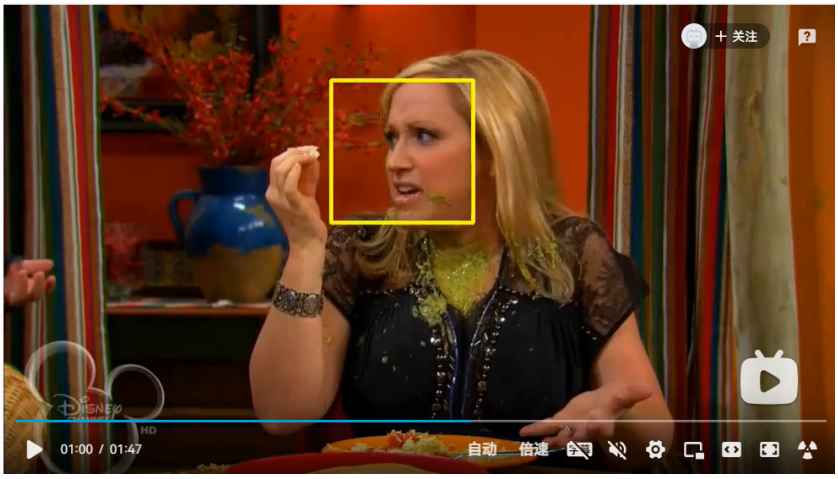

Fig. 3.15. screenshot 1

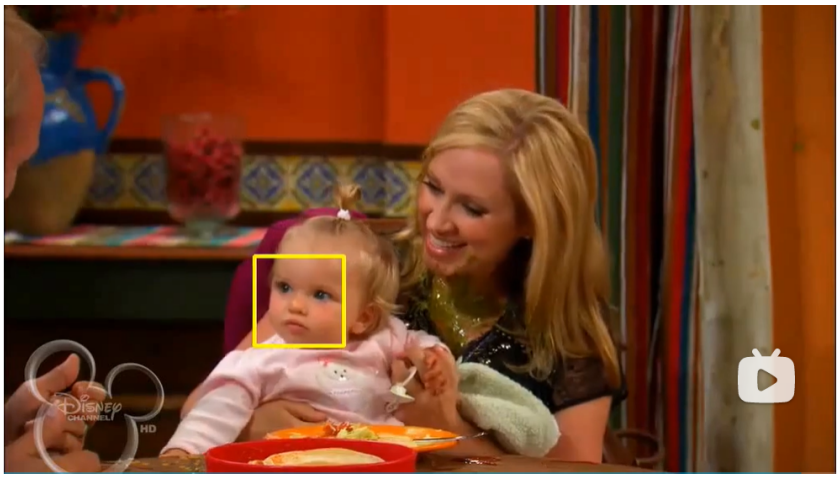

Fig. 3.16. screenshot 2

1.c full source code

You can click ⋆here⋆ to see the full code for this part.

2.Face Detection in Video Streams.(using Web-camera)

1.a Image processing result

The following is a screenshot of the program running process:

1.c full source code

You can click ⋆here⋆ to see the full code for this part.

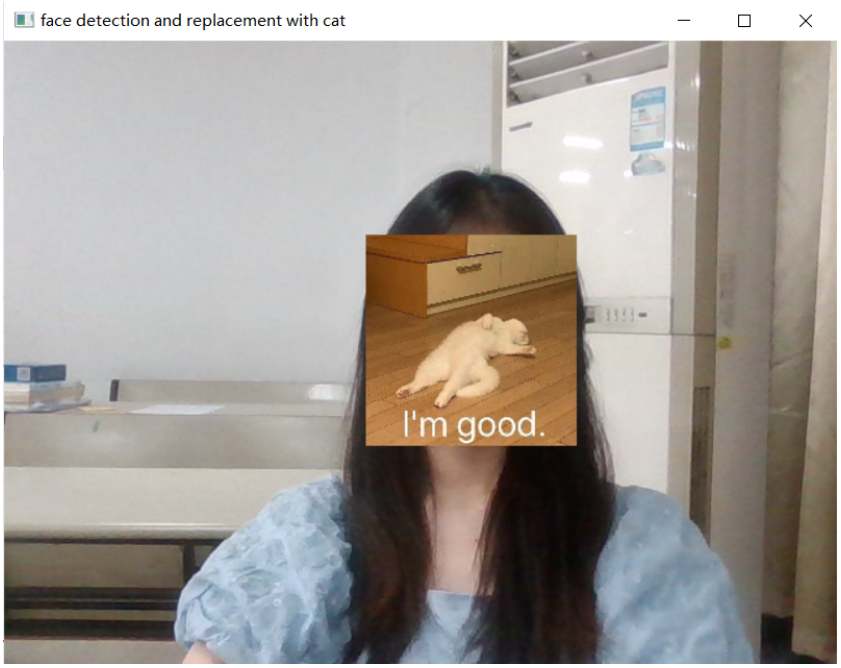

Fig. 3.17. Screenshot of video result

3.Comments

- ★ The face detection algorithm in the video needs to be improved. From the experimental results, there are many false positive detection results. This is because unlike images, we can adjust the parameters for each photo. There is only one preset parameter in the video, and this parameter may not be suitable for every frame of the video.

4 Conclusion

In this exercise assignment, I learned the following application methods:

- ★ The Viola-Jones face detector method is based on the following concepts:
  1) **Haar-like feaures** as **weak classifiers**.
  2) **Integral image** representation for **fast calculation** of Haar-like features.
  3) A**daBoost training method** to **combine weak classifier**s into a **strong classifier**.
  4) Combining of strong classifiers into a **cascade classifier.**
- ★ Sometimes in object detection, we don't need to scan the entire image.False positive results can be reduced by detecting **Region-Of-Interest (ROI)** first and then performing subsequent detections
- ★ The essence of video is a **video stream formed by frame by frame images.** When we process the video, it can be split into frame by frame images for processing and then stitched together.

Note: The characteristics of each method are described in the previous comments and will not be repeated here

5 The answer to the questions

1) What is the special image representation used in the Viola-Jones approach?

Sometimes in object detection, we don't need to scan the whole image (for example, when we examine the eyes, we can scan only the face image), which will avoid many false positive results. So we define a **Region-Of-Interest (ROI)** that has been found, and then search for objects under the Region-Of-Interest (ROI). (For example, for face area images that search for eyes).
This ROI image is the special image used in the Viola-Jones method.

2) What is the main advantage of Haar-like features for classifier training?

The main advantage of the Haar-like features is that it is **very fast to compute**. As shown in the previous theory, Haar-like features of arbitrary size can be computed in constant time using a structure called an integral image.

3) Could you use Viola-Jones approach for detecting arbitrary objects and why?

4) How to use feature points for stitching a panoramic image?

The Viola-Jones approach **can be trained to detect various object categories**, but its **main** motivation is the **face detection** problem.
For detection of different objects we can train different cascade descriptors for detection. However, it should be noted that **whether** object detection can achieve **good results requires experiments to illustrate**. From the previous experiments, it can be seen that the algorithm can be used to detect frontal face images, but it is not very robust for side face image detection.

I

Appendix

A Complete source code

§A.1 Faces detection.

and you can click ⋆here⋆ to return to reading the report

Python

```
import cv2
import matplotlib.pyplot as plt

I = cv2.imread('good luck.jpg')
#I = cv2.imread('bxqy.png')
#I = cv2.imread('hlbt.png')
Igray = cv2.cvtColor(I,cv2.COLOR_BGR2GRAY)

detector = cv2.CascadeClassifier()
cascade_fn = cv2.samples.findFile("haarcascade_frontalface_default.xml")
detector.load(cascade_fn)
faces = detector.detectMultiScale(Igray, scaleFactor = 1.07, minNeighbors = 12)
Iout = I.copy()
for (x,y,w,h) in faces:
    Iout = cv2.rectangle(Iout,(x,y,w,h),(0,255,255),3)

I = cv2.cvtColor(I,cv2.COLOR_BGR2RGB)
Iout = cv2.cvtColor(Iout,cv2.COLOR_BGR2RGB)
plt.imshow(I),plt.title('original image with faces'),plt.axis ('off')
plt.show()
plt.waitforbuttonpress

plt.imshow(Iout),plt.title(' Image with detected faces highlighted'), plt.axis ('off')
plt.show()
plt.waitforbuttonpress
```

§A.2 Body parts detectiong.

and you can click ⋆here⋆ to return to reading the report

Python

```
import cv2
import matplotlib.pyplot as plt
```

```
3
4  #I = cv2.imread('good luck.jpg')
5  I = cv2.imread('bxqy.png')
6  #I = cv2.imread('hlbt.png')
7  Igray = cv2.cvtColor(I,cv2.COLOR_BGR2GRAY)
8
9  detector_face = cv2.CascadeClassifier()
10 cascade_face = cv2.samples.findFile("haarcascade_frontalface_default.
       xml")
11 detector_face.load(cascade_face)
12 detector_eye = cv2.CascadeClassifier()
13 cascade_eye = cv2.samples.findFile("haarcascade_eye.xml")
14 detector_eye.load(cascade_eye)
15 detector_mouth = cv2.CascadeClassifier()
16 cascade_mouth = cv2.samples.findFile("haarcascade_mcs_mouth.xml")
17 detector_mouth.load(cascade_mouth)
18 detector_nose = cv2.CascadeClassifier()
19 cascade_nose = cv2.samples.findFile("haarcascade_mcs_nose.xml")
20 detector_nose.load(cascade_nose)
21 faces = detector_face.detectMultiScale(Igray, scaleFactor = 1.07,
       minNeighbors = 12)
22 Iout = I.copy()
23 for (x,y,w,h) in faces:
24     Iout = cv2.rectangle(Iout,(x,y,w,h),(0,255,255),3)
25     Iface = Igray[y:y+h,x:x+w]
26     Iface_top = Igray[y:y+h*2 // 3,x:x+w]
27     Iface_bottom = Igray[y+(h//5*3):y+h,x:x+w]
28     Iface_mid = Igray[y+(h//4):y+(h//4*3),x:x+w]
29     eyes = detector_eye.detectMultiScale (Iface_top, scaleFactor =
           1.05, minNeighbors = 1)
30     mouths = detector_mouth.detectMultiScale (Iface_bottom, scaleFactor
           = 1.05, minNeighbors = 1)
31     noses = detector_nose.detectMultiScale (Iface_mid, scaleFactor =
           1.05, minNeighbors = 1)
32     for (x2,y2,w2,h2)in eyes:
33         Iout = cv2.rectangle(Iout,(x+x2,y+y2,w2,h2),(0,255,0),3)
34     for (x3,y3,w3,h3)in mouths:
35         Iout = cv2.rectangle(Iout,(x+x3,y+(h//5*3)+y3,w3,h3),(255,0,0)
               ,3)
36     for (x4,y4,w4,h4)in noses:
37         Iout = cv2.rectangle(Iout,(x+x4,y+(h//4)+y4,w4,h4),(0,0,255),3)
38 I = cv2.cvtColor(I,cv2.COLOR_BGR2RGB)
39 Iout = cv2.cvtColor(Iout,cv2.COLOR_BGR2RGB)
40 plt.imshow(I),plt.title('original image with faces'),plt.axis ('off')
41 plt.show()
42 plt.waitforbuttonpress
43
44 plt.imshow(Iout),plt.title(' Image with detected faces highlighted'),
       plt.axis ('off')
45 plt.show()
```

```
plt.waitforbuttonpress
```

§A.3 Optional 1.face detection in videostream(pre-recorded video)

and you can click ⋆here⋆ to return to reading the report

Problem 1

```
import cv2
from moviepy import *
from moviepy.editor import *
def op_one_img(I):
    Igray = cv2.cvtColor(I,cv2.COLOR_BGR2GRAY)
    detector = cv2.CascadeClassifier()
    cascade_fn = cv2.samples.findFile("haarcascade_frontalface_default.xml")
    detector.load(cascade_fn)
    faces = detector.detectMultiScale(Igray, scaleFactor = 1.07, minNeighbors = 15)
    Iout = I.copy()
    for (x,y,w,h) in faces:
        Iout = cv2.rectangle(Iout,(x,y,w,h),(0,255,255),3)
    return Iout


def makevideo():
    videoinpath  = 'test.avi'
    videooutpath = 'test_out.avi'
    capture     = cv2.VideoCapture(videoinpath  )
    fps = capture.get(cv2.CAP_PROP_FPS)
    fourcc      = cv2.VideoWriter_fourcc(*'XVID')
    writer      = cv2.VideoWriter(videooutpath ,fourcc, fps, (1024,576), True)
    if capture.isOpened():
        while True:
            ret,img_src=capture.read()
            if not ret:break
            img_out = op_one_img(img_src)
            # ?????op_one_img()????
            writer.write(img_out)
    else:
        print('????????')
    writer.release()

makevideo()

video = VideoFileClip('test.avi')
```

```
video_out = VideoFileClip('test_out.avi')
audio = video.audio
#audio.write_audiofile('audio.mp3')
videoclip = video_out.set_audio(audio)
videoclip.write_videofile('test_out1.mp4')
```

§A.4 Optional 1.face detection in videostream(using web-camera)

and you can click ⋆here⋆ to return to reading the report

Python

```
import cv2
face_detector = cv2.CascadeClassifier()
cascade_fn = cv2.samples.findFile('./haarcascade_frontalface_alt.xml')
face_detector.load(cascade_fn)
# Get camera behavior
cap = cv2.VideoCapture(0)
while True:
    # Returns pictures by frame from the camera
    flag,frame = cap.read()
    if not flag :
        break
    frame_gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(frame_gray, scaleFactor =
       1.07, minNeighbors = 12)
    for (x,y,w,h) in faces:
        cat = cv2.imread('./head.jpg')
        cat = cv2.resize(cat,dsize = (w,h))
        frame[y:y+h,x:x+w] = cat
    cv2.imshow('face detection and replacement with cat',frame)
    key = cv2.waitKey(10)
    if key == ord('q'): # Enter q to quit reading
            break
cv2.destroyAllWindows()
cap.release()
```