**Computer vision – HW10**

[Question 1 2](#_Toc152018013)

[A、 Source code 2](#_Toc152018014)

[B、 Analyze 3](#_Toc152018015)

[C、 Result map 5](#_Toc152018016)

[Question 2 6](#_Toc152018017)

[A、 Source code 6](#_Toc152018018)

[B、 Result map 8](#_Toc152018019)

|  |  |
| --- | --- |
| Name: | WeiWen Wu |
| Student ID: | 10901007A |

# Question 1

## Source code

|  |
| --- |
| **from** numpy **import** ndarray  **import** cv2  **from** typing **import** Optional  FONT**:int** **=** cv2**.**FONT\_HERSHEY\_SIMPLEX  **def** bgr2gray**(**img**:**ndarray**)** **->** ndarray**:**  """Convert bgr to gray and remove noise points."""  **return** cv2**.**equalizeHist**(**cv2**.**cvtColor**(**img**,** cv2**.**COLOR\_BGR2GRAY**))** # BGR to gray.  **def** plot\_from\_cascade**(**img**:**ndarray**,**object\_name**:str,object:**ndarray**)** **->** Optional**[tuple[**ndarray**,**ndarray**]]:**  """Read cascade data and draw pictures."""  **if** **len(object)==**0**:** **return** **None**  **for** **(**x**,** y**,** w**,** h**)** **in** **object:**  cv2**.**rectangle**(**img**,** **(**x**,** y**),** **(**x**+**w**,** y**+**h**),** **(**255**,** 0**,** 0**),** 2**)**  roi\_gray **=** gray**[**y**:**y**+**h**,** x**:**x**+**w**]**  roi\_color **=** img**[**y**:**y**+**h**,** x**:**x**+**w**]**  cv2**.**putText**(**img**,** object\_name **,** **(**x**,** y**),** FONT**,** .5**,** **(**0**,** 255**,** 255**),** 2**)**  **return** roi\_gray**,**roi\_color  folder **=** "./HW10/Haar cascade classifier xml files"  object\_cascade **=** cv2**.**CascadeClassifier**(**f'{folder}/haarcascade\_smile.xml'**)**  cap **=** cv2**.**VideoCapture**(**0**)**  cv2**.**namedWindow**(**"cascade"**)**  ### Create a track bar ### cv2.createTrackbar('Slider name', 'window name', min, max, fn)  cv2**.**createTrackbar**(**"scaleFactor"**,** "cascade"**,** 150**,** 500**,** **lambda** \_**:**\_**)**  cv2**.**createTrackbar**(**"minNeighbors"**,** "cascade"**,** 0**,** 20**,** **lambda** \_**:**\_**)**  cv2**.**setTrackbarPos**(**"scaleFactor"**,** "cascade"**,** 400**)**  cv2**.**setTrackbarPos**(**"minNeighbors"**,** "cascade"**,** 18**)**  **while** **True:**  ret**,** img **=** cap**.**read**()**  gray **=** bgr2gray**(**img**)**  ### Get trackbar position. ### cv2.setTrackbarPos('Slider name', 'window name', default)  scaleFactor**=** cv2**.**getTrackbarPos**(**'scaleFactor'**,**'cascade'**)**  minNeighbors**=** cv2**.**getTrackbarPos**(**'minNeighbors'**,**'cascade'**)**  smile **=** object\_cascade**.**detectMultiScale**(**gray**,** scaleFactor**/**100**,** minNeighbors**)**  plot\_from\_cascade**(**img**,**"smile"**,**smile**)**  cv2**.**imshow**(**'cascade'**,** img**)**  k **=** cv2**.**waitKey**(**30**)** **&** 0xff  **if** k **==** 27**:**  **break**  cap**.**release**()**  cv2**.**destroyAllWindows**()** |

## Analyze

原來的圖片，如Figure 1所示，使用Trackbar來調整detectMultiScale中scaleFactor與minNeighbors之值。

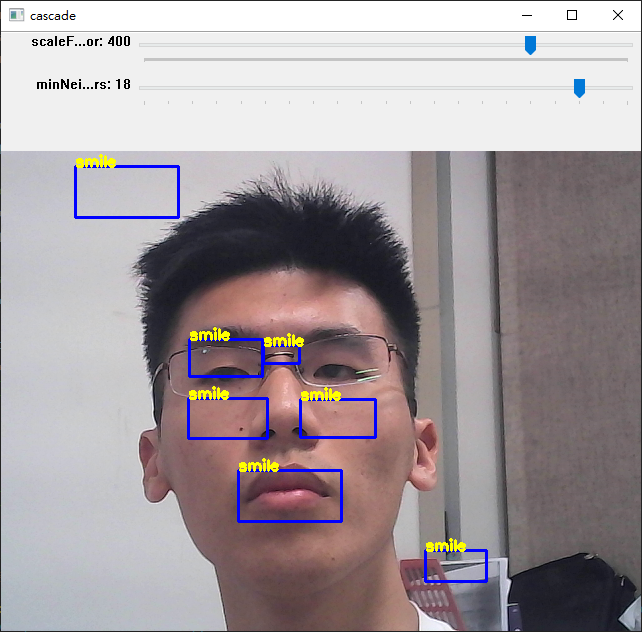
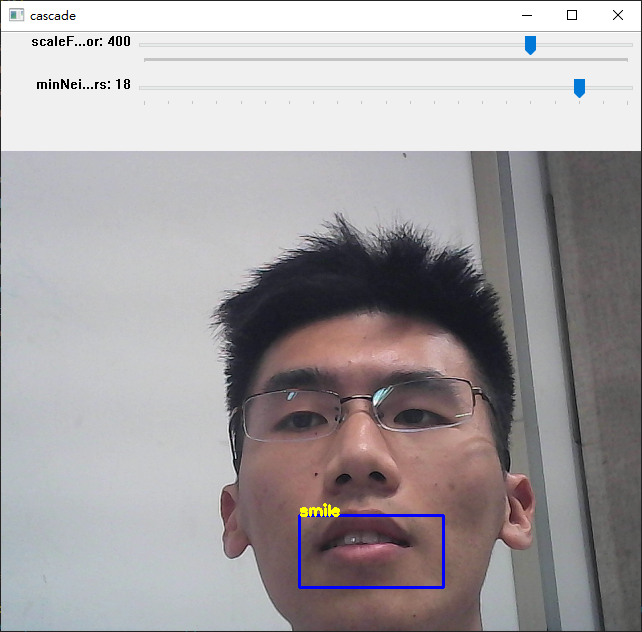


Figure 1 Original picture

## Result map

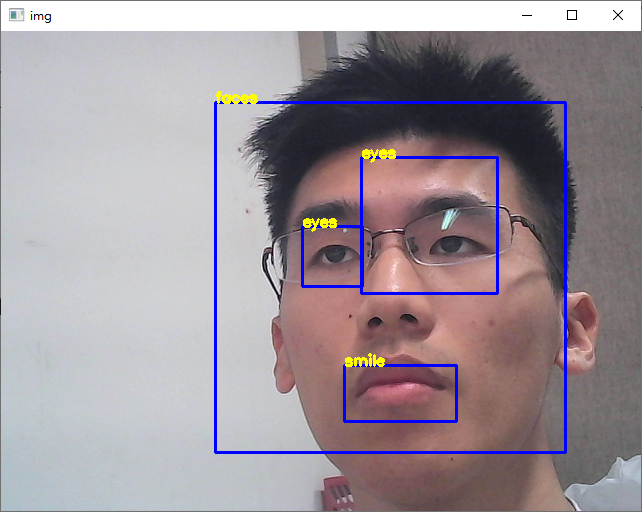


# Question 2

## Source code

|  |
| --- |
| **from** numpy **import** ndarray  **import** cv2  **from** typing **import** Optional  FONT **=** cv2**.**FONT\_HERSHEY\_SIMPLEX  **def** bgr2gray**(**img**:**ndarray**)** **->** ndarray**:**  """Convert bgr to gray and equalizeHist."""  **return** cv2**.**equalizeHist**(**cv2**.**cvtColor**(**img**,** cv2**.**COLOR\_BGR2GRAY**))** # BGR to gray.  **def** plot\_from\_cascade**(**img**:**ndarray**,**object\_name**:str,object:**ndarray**)** **->** Optional**[tuple[**ndarray**,**ndarray**]]:**  """Read cascade data and draw pictures."""  **if** **len(object)==**0**:** **return** **None**  **for** **(**x**,** y**,** w**,** h**)** **in** **object:**  cv2**.**rectangle**(**img**,** **(**x**,** y**),** **(**x**+**w**,** y**+**h**),** **(**255**,** 0**,** 0**),** 2**)**  roi\_gray **=** gray**[**y**:**y**+**h**,** x**:**x**+**w**]**  roi\_color **=** img**[**y**:**y**+**h**,** x**:**x**+**w**]**  cv2**.**putText**(**img**,** object\_name **,** **(**x**,** y**),** FONT**,** .5**,** **(**0**,** 255**,** 255**),** 2**)**  **return** roi\_gray**,**roi\_color  cap **=** cv2**.**VideoCapture**(**0**)**  ### Import identification tools. ###  folder **=** "./HW10/Haar cascade classifier xml files"  face\_cascade **=** cv2**.**CascadeClassifier**(**f'{folder}/haarcascade\_frontalface\_default.xml'**)**  eye\_cascade **=** cv2**.**CascadeClassifier**(**f'{folder}/haarcascade\_eye.xml'**)**  smile\_cascade **=** cv2**.**CascadeClassifier**(**f'{folder}/haarcascade\_smile.xml'**)**  profileface\_cascade **=** cv2**.**CascadeClassifier**(**f'{folder}/haarcascade\_profileface.xml'**)**  cv2**.**namedWindow**(**"cascade"**)**  ### Create a track bar ###  ## cv2.createTrackbar('Slider name', 'window name', min, max, fn)  cv2**.**createTrackbar**(**"scaleFactor"**,** "cascade"**,** 150**,** 500**,** **lambda** \_**:**\_**)**  cv2**.**createTrackbar**(**"minNeighbors"**,** "cascade"**,** 0**,** 20**,** **lambda** \_**:**\_**)**  cv2**.**setTrackbarPos**(**"scaleFactor"**,** "cascade"**,** 107**)**  cv2**.**setTrackbarPos**(**"minNeighbors"**,** "cascade"**,** 1**)**  **while** 1**:**  ret**,** img **=** cap**.**read**()**  gray **=** bgr2gray**(**img**)**  # detectMultiScale(roi\_gray, scaleFactor=1.02, minNeighbors=3, minSize=(40,40))  faces **=** face\_cascade**.**detectMultiScale**(**gray**,** 1.3**,** 5**)**  profileface **=** profileface\_cascade**.**detectMultiScale**(**gray**,** 1.3**,** 5**)**  ### Read cascade data and draw pictures. ###  roi **=** plot\_from\_cascade**(**img**,**"faces"**,**faces**)**  plot\_from\_cascade**(**img**,**"profileface"**,**profileface**)**    ### If no face is detected, eyes and mouth will not be detected. ###  **if** roi **!=** **None:**  roi\_gray**,**roi\_color **=** roi  ### Get trackbar position. ###  scaleFactor**=** cv2**.**getTrackbarPos**(**'scaleFactor'**,**'cascade'**)**  minNeighbors**=** cv2**.**getTrackbarPos**(**'minNeighbors'**,**'cascade'**)**  # detectMultiScale(roi\_gray, scaleFactor=1.02, minNeighbors=3, minSize=(40,40))  eyes **=** eye\_cascade**.**detectMultiScale**(**roi\_gray**,** 1.1**,** 1**)**  smile **=** smile\_cascade**.**detectMultiScale**(**roi\_gray**,** 3.1**,** 19**)**  ### Read cascade data and draw pictures. ###  plot\_from\_cascade**(**roi\_color**,**"eyes"**,**eyes**)**  plot\_from\_cascade**(**roi\_color**,**"smile"**,**smile**)**  cv2**.**imshow**(**'img'**,** img**)**  k **=** cv2**.**waitKey**(**30**)** **&** 0xff  **if** k **==** 27**:**  **break**    cap**.**release**()**  cv2**.**destroyAllWindows**()** |

## Result map



# 