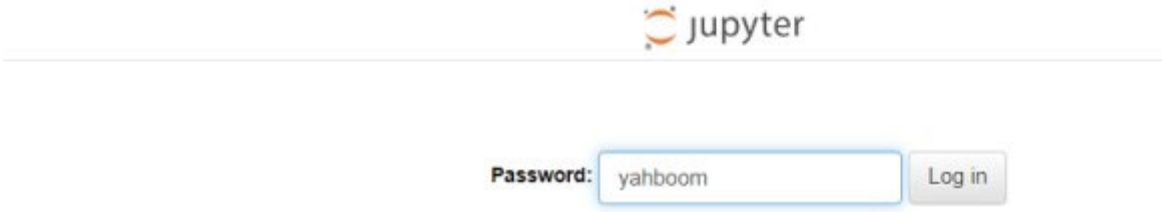
Face Tracking

1. Purpose of the experiment

This tutorial will fully introduce how to deploy a face detection and recognition system for a robot dog, capture facial features in real time through OpenCV and deep learning technology, calculate the target position coordinates, and use control algorithms to adjust the robot dog's motion posture to achieve stable human tracking function.

2. Main source code path

First, end the big program, then open the browser and enter "ip (ip is the ip of the robot dog): 8888", enter the password "yahboom" and enter

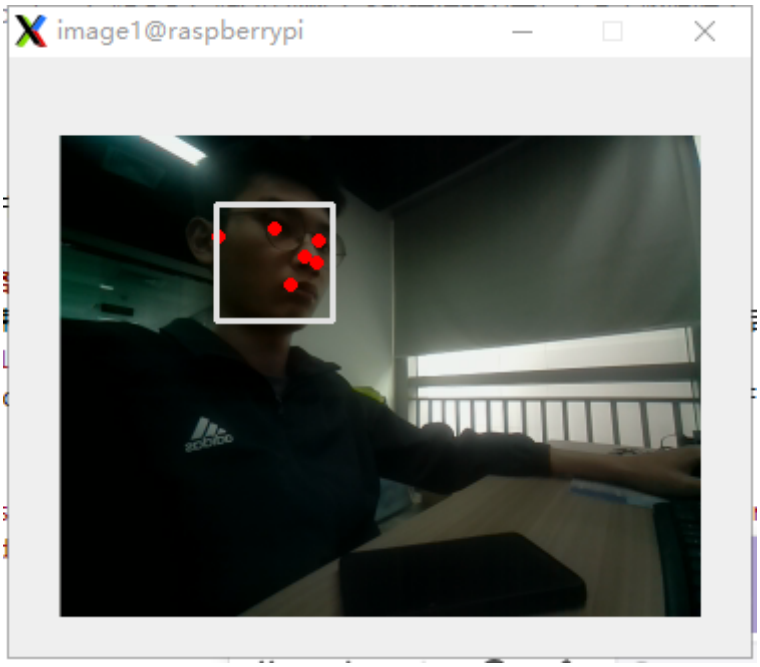


the path to ~/DOGZILLA_Lite_class/6.AI Visual Interaction Course/07.Facial tracking. Open the **face_decetion.ipynb** program and run it ，or enter it in the terminal

```
cd ~/DOGZILLA_Lite_class/6.AI Visual Interaction Course/07.Facial tracking
python3 face_decetion.py
```

3. Experimental Phenomenon

After running the source code, the robot dog's body will follow the recognized face and track it.



4. Main source code analysis

```
try:
  with mp_face_detection.FaceDetection(
      model_selection=0, min_detection_confidence=0.5) as face_detection:
    while cap.isOpened():
      success, image = cap.read()
      if not success:
        print("Ignoring empty camera frame.")
        # If loading a video, use 'break' instead of 'continue'.
        continue

      # To improve performance, optionally mark the image as not writeable to
      # pass by reference.
      image.flags.writeable = False
      image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
      results = face_detection.process(image)

      # Draw the face detection annotations on the image.
      image.flags.writeable = True
      image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
      if results.detections:
        for detection in results.detections:
          value_x=0
          value_y=0
          mp_drawing.draw_detection(image, detection)
          xy=(mp_face_detection.get_key_point(detection,
mp_face_detection.FaceKeyPoint.NOSE_TIP))
          face_x=320-xy.x*320
          face_y=xy.y*240
          value_x = face_x - 160
          value_y = face_y - 120
          rider_x=value_x
          print(face_x,face_y)
          if value_x > 55:
            value_x = 55
          elif value_x < -55:
            value_x = -55
          if value_y > 75:
            value_y = 75
          elif value_y < -75:
            value_y = -75

      else:
        value_x=value_y=face_x=face_y=0
        rider_x=9999

      print(['y','p'],[value_x/9, value_y/15])
      g_dog.attitude(['y','p'],[value_x/9, value_y/15])

      b,g,r = cv2.split(image)
      image = cv2.merge((r,g,b))
      image = cv2.flip(image, 1)
      imgok = Image.fromarray(image)
      display.ShowImage(imgok)
```

```
        #显示在屏幕上 Display on the screen
        r,g,b = cv2.split(image)
        image1 = cv2.merge((b,g,r))
        cv2.imshow("image1",image1)

        # Flip the image horizontally for a selfie-view display.
        #cv2.imshow('MediaPipe Face Detection', cv2.flip(image, 1))
        if cv2.waitKey(5) & 0xFF == 27:
          break
        if button.press_b():
          g_dog.reset()
          break
except:
  cap.release()
  g_dog.reset()
```

The above source code is only the tracking control part. Press the button on the lower left of the screen to exit recognition and end this case.