Watchdog

1. Introduction

This course mainly uses the camera of Raspberry Pi to obtain the picture of the camera. The opencv library imports the face Haar feature classifier to analyze the position of the face in the image and frame the face. If the face image is within the detection range and the size is greater than the set threshold, the handshake action is executed.

2. Code analysis

Importing face Haar feature classifier

```
# 导入人脸Haar特征分类器  Import face Haar feature classifier
face_haar = cv2.CascadeClassifier("haarcascade_profileface.xml")
```

In the task of greeting the watchdog, the camera image is grayed first, and then the image is sent to the Haar face detection feature classifier for processing. The face is detected and the data of the face position is output and draw a box for the face in the image, and then transfer the data to the action processing function action_ Handle.

```
# 任务：检测到人脸执行握手动作 Task: Execute hanshake action when face is detected
def Face_Greet_Task():
    count = 0
    action_time = 0
    t_start = time.time()
    fps = 0
    while True:
        ret, frame = image.read()
        # 把图像转为灰度图像  Convert the image to grayscale
        gray_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_haar.detectMultiScale(gray_img, 1.1, 3)
        for face_x,face_y,face_w,face_h in faces:
            cv2.rectangle(frame, (face_x, face_y), (face_x+face_w, face_y+face_h), (0,255,0), 2)
            action_handle(face_x+face_w/2,face_y+face_h/2,face_w,face_h)
        fps = fps + 1
        mfps = fps / (time.time() - t_start)
        cv2.putText(frame, "FPS " + str(int(mfps)), (40,40), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        image_widget.value = bgr8_to_jpeg(frame)
```

action_handle determines whether it meets the handshake action requirements according to the image data.

```
def action_handle(x, y, w, h):
    if w > 60 and h > 60 and (150 <= x <= 450) and (100 <= y <= 380):
        g_dog.action(19) # 握手 Handshake
        time.sleep(.1)
```

Start a daemon thread, run the camera identification task, and display the camera image.

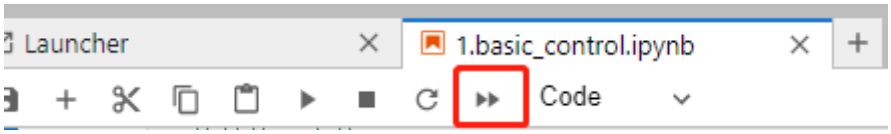
```
# 启动摄像头显示任务  Start the camera display task
thread1 = threading.Thread(target=Face_Greet_Task)
thread1.setDaemon(True)
thread1.start()

box_display = widgets.HBox([image_widget, button_Close_Camera])
display(box_display)
```

1.3 Steps

Open the jupyterLab client and find following code path

```
DOGZILLA/Samples/3_AI_Visual/6.face_handshake.ipynb
```

Click the following icon to run all cells, and then pull to the bottom to see the generated controls.



Please place the robot dog in a pure color environment of the camera image, otherwise it may cause false recognition due to color interference.

The left side shows the camera image. When the face is moved in front of the camera, the image will draw a box for the face. If the length and width of the box are greater than 60 and the center point is within the detection range in the middle of the image, the robot dog will perform handshake.

Finally, click close_camera button turns off the camera.