import cv2 # 計算機視覺和圖像處理  
import numpy as np # 科學計算和數組處理  
import serial  
import time  
  
ser = serial.Serial('COM4', baudrate=115200, timeout=1)  
time.sleep(0.5)  
pos = 90  
pos2 = 90  
  
confThreshold = 0.2 # 設置物體檢測的置信度閾值，只有置信度分數大於閾值的檢測結果才會被保留。  
# 閾值的作用是過濾掉置信度分數低於閾值的檢測結果，以減少錯誤檢測和冗余檢測  
  
# 初始化第一個網絡攝像頭  
cam = cv2.VideoCapture(0, cv2.CAP\_DSHOW)  
  
# 讀取 coco80.names 文件以獲取類別名稱  
classesFile = 'Resources/coco80.names'  
  
# 定義一個空列表用於存儲從 classesFile 文件中讀取的物體類別名稱  
classes = []  
  
# 打開 classesFile 文件  
with open(classesFile, 'r') as f:  
  
 # read() 方法讀取文件內容  
 # splitlines() 方法將文件內容分割成一個個單獨的行，並將它們存儲在 classes 列表中。  
 classes = f.read().splitlines()  
  
 # 列印出 classes 列表和列表的長度  
 print("coco80 類別 : " , classes)  
 print("coco80 類別長度 : " ,len(classes))  
  
# 使用 OpenCV 的 dnn 模組中的 readNetFromDarknet() 函數從 Darknet 框架的配置文件和權重文件中讀取 YOLOv3 模型。  
net = cv2.dnn.readNetFromDarknet('Resources/yolov3.cfg', 'Resources/yolov3.weights')  
  
  
# 使用 OpenCV 作為後端  
net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_OPENCV)  
# 使用 CPU 作為目標設備來運行 YOLOv3 模型  
net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CPU)  
  
time.sleep(5)  
  
while True:  
  
 # 從攝像頭中讀取一幀圖像。這個方法返回兩個值：  
 # 一個布爾值 success : 表示圖像是否成功讀取，如果讀取成功，則 success 為 True，否則為 False。  
 # 一個圖像 img : 一個 NumPy 數組，它包含從攝像頭中讀取的圖像數據。  
 success , img = cam.read()  
  
 # 獲取圖像的高度、寬度和通道數。  
 height, width, ch = img.shape  
  
 # 使用 OpenCV 的 dnn 模組中的 blobFromImage() 函數將圖像轉換成一個二進制大型物件（blob）  
 # blobFromImage() 函數 :  
 # img ： 要轉換的圖像。  
 # 1 / 255 ： 像素值縮放係數，將像素值從 0 到 255 縮放到 0 到 1 之間。  
 # (320, 320) ： 輸出的 blob 的尺寸。YOLOv3 模型需要輸入一個固定大小的圖像，因此需要將圖像調整為指定的大小。  
 # (0, 0, 0) ： 均值，減去這個均值可以對圖像進行標準化。  
 # swapRB=True： 交換通道，將圖像通道的順序從 BGR 轉換為 RGB。  
 # crop=False ： 是否進行裁剪，如果裁剪，則圖像的長寬比將被保持不變，並進行裁剪。  
 # 返回一個 NumPy 數組，這個數組包含了轉換後的 blob 數據。  
 blob = cv2.dnn.blobFromImage(img, 1 / 255, (320, 320), (0, 0, 0), swapRB=True, crop=False)  
  
 # 將剛才轉換成的 blob 數據設置為 YOLOv3 模型的輸入  
 net.setInput(blob)  
  
 # net.getLayerNames() 方法返回一個列表，這個列表包含了 YOLOv3 模型中卷積層、池化層、激活層等所有層的名稱  
 layerNames = net.getLayerNames()  
  
 # 列印出 YOLOv3 模型中的所有層的名稱。  
 #　print("YOLOv3 模型中的所有層層 YOLOv3 Layers : ")  
 # print(layerNames)  
  
 # net.getUnconnectedOutLayersNames() 方法獲取 YOLOv3 模型的輸出層的名稱。  
 output\_layers\_names = net.getUnconnectedOutLayersNames()  
  
 # 列印出 YOLOv3 模型中的輸出層的名稱。  
 # print("YOLOv3 模型中的輸出層 YOLOv3 Output Layers : ")  
 # print(output\_layers\_names)  
  
 # 使用 YOLOv3 模型的 forward() 方法進行前向計算，並返回一個包含了模型輸出結果的列表。  
 LayerOutputs = net.forward(output\_layers\_names)  
  
 # 列印出模型輸出結果列表的長度  
 #　print("模型輸出結果列表的長度 : ",len(LayerOutputs))  
 # print(LayerOutputs[0].shape)  
 # print(LayerOutputs[1].shape)  
 # print(LayerOutputs[2].shape)  
 # print(LayerOutputs[0][0])  
  
 # 列表 : 用於存儲所有檢測到的物體的邊界框  
 bboxes = []  
 # 列表 : 存儲所有檢測到的物體的置信度  
 confidences = []   
 # 列表 : 存儲所有檢測到的物體的類別 ID  
 class\_ids = []  
  
 # 遍歷所有模型輸出層的輸出結果  
 for output in LayerOutputs:  
 # print("Output : ")  
 # print(output)  
 # 遍歷輸出結果所檢測到的物體  
 for detection in output:  
 # print("detection : ")  
 # print(detection)  
  
 # 提取 detection 中的所有置信度值(物體屬於每個類別的置信度)   
 scores = detection[5:]  
 # print("scores: ",scores)  
 # 使用 NumPy 的 argmax() 函數找到 scores 中置信度最高的類別，並將其賦值給變量 class\_id  
 class\_id = np.argmax(scores)  
 # 提取物體屬於class\_id的置信度  
 confidence = scores[class\_id]  
 # print("confidence", confidence)  
 # 判斷檢測到的物體的置信度是否大於預設的置信度閾值  
 if confidence > confThreshold:  
 # 檢測到的物體的邊界框中心點在圖像中的位置，並將其賦值給變量 center\_x 和 center\_y。  
 #　detection[0] 和 detection[1] 分別提取出檢測到的物體邊界框的中心點在圖像中的比例座標  
 # 將其乘以圖像的寬和高，得到在圖像中的真實座標值。  
 center\_x = int(detection[0]\*width)  
 center\_y = int(detection[1]\*height)  
  
 # detection[2] 和 detection[3] 分別提取出檢測到的物體邊界框的寬和高在圖像中的比例  
 #　然後將其乘以圖像的寬和高，得到在圖像中的真實座標值。  
 w = int(detection[2]\*width)  
 h = int(detection[3]\*height)  
  
 #　計算出邊界框的左上角座標 x 和 y。  
 x = int(center\_x - w/2)  
 y = int(center\_y - h/2)  
  
 # 將計算出的邊界框位置和大小資訊加入到 bboxes 列表中。  
 bboxes.append([x,y,w,h])  
 # print(bboxes)  
 # 將檢測到的物體的置信度值轉換成浮點數型，然後將其加入到 confidences 列表中。  
 confidences.append((float(confidence)))  
 # 將檢測到的物體的類別編號加入到 class\_ids 列表中。  
 class\_ids.append(class\_id)  
 # cv2.rectangle(img, (x, y), (x + w, y + h), (255,0,0), 2)  
 # print("<<<detection")  
 # 使用非極大值抑制（Non-maximum Suppression，NMS）方法對檢測結果進行後處理，以過濾掉重疊的邊界框和低置信度的檢測結果。  
 # bboxes：包含所有檢測到的物體邊界框位置和大小的列表；  
 # confidences：包含所有檢測到的物體的置信度值的列表；  
 # confThreshold：置信度閾值，低於此閾值的檢測結果將被過濾掉；  
 # nmsThreshold：NMS 閾值，用於過濾掉重疊的邊界框。  
 # 輸出結果是一個包含選擇後的檢測結果索引的列表 indexes，這些索引對應的檢測結果為最佳的、不重疊的、置信度最高的邊界框。  
 indexes = cv2.dnn.NMSBoxes(bboxes, confidences, confThreshold, 0.4) # Non-maximum suppresion  
 # print(indexes)  
 # print(indexes.flatten())  
  
 # 定義字型，這裡使用的是 OpenCV 的字型，指定為平面字型。  
 font = cv2.FONT\_HERSHEY\_PLAIN  
 # 使用 NumPy 的 random 函式，產生一個由亂數組成的 Numpy 陣列，數字範圍在 0 到 255 之間。  
 # 陣列形狀為 (len(bboxes), 3)，其中 len(bboxes) 是圖像中邊框的數量，3 代表 RGB 三個通道的顏色。   
 # 這樣可以為圖像中的每個邊框分配一個獨特的顏色，用於繪製每個檢測到的對象的邊框或將每個對象標記為獨特的顏色。  
 colors = np.random.uniform(0,255,size=(len(bboxes),3))  
  
 # 如果 indexes 的長度大於 0，代表圖像中有檢測到物體，進入迴圈：  
 if len(indexes) > 0:  
  
 # indexes.flatten() 將 indexes 陣列展開成一維陣列  
 # 包含所有選擇後的物體的索引  
 for i in indexes.flatten():  
  
 # 取出對應的邊框座標 x、y、w、h，以及對應的類別標籤 label 和信心分數 confidence  
 x,y,w,h = bboxes[i]  
 label = str(classes[class\_ids[i]])  
 confidence = str(round(confidences[i],2))  
  
 if label == 'mouse':  
  
 # 透過 colors 陣列取得第 i 個邊框對應的顏色值  
 color = colors[i]  
  
 # 使用 OpenCV 的 cv2.rectangle 函式在圖像上繪製邊框，指定邊框顏色和線條粗細。  
 cv2.rectangle(img,(x,y),(x+w,y+h),color,2)  
  
 # 使用 cv2.putText 函式在圖像上標註物體的類別名稱和信心分數  
 cv2.putText(img,label+" "+ confidence,(x,y+20),font,2,(255,255,255),2)  
  
 errorPan = (x + w/2) - 640/2 # 计算舵机与人脸中心的偏差  
 errorPan2 = (y + h / 2) - 480 / 2  
 print('errorPan', errorPan)  
  
 print('Pos', pos)  
 if abs(errorPan) > 40: # 只有偏差大于一定值时才进行舵机控制  
 pos = pos - errorPan/40 # 根据偏差调整舵机位置  
 if pos > 160: # 如果舵机位置超出范围，则将其限制在范围内  
 pos = 160  
 print("超出范围")  
 elif pos < 0:  
 pos = 0  
 print("超出范围")  
  
 if abs(errorPan2) > 40: # 只有偏差大于一定值时才进行舵机控制  
 pos2 = pos2 + errorPan2 / 40 # 根据偏差调整舵机位置  
 if pos2 > 160: # 如果舵机位置超出范围，则将其限制在范围内  
 pos2 = 160  
 print("超出范围")  
 elif pos2 < 0:  
 pos2 = 0  
 print("超出范围")  
  
 servoPos = str(int(pos)) + ","+str(int(pos2)) + '\r' # 将舵机位置转换为字符串并发送给舵机  
 ser.write(servoPos.encode('utf-8'))  
 print('舵机位置 = ', servoPos)  
  
 # 彈出視窗並顯示圖像。  
 cv2.imshow('Image', img)  
 # key = cv2.waitKey(1)  
 # if key == 27:  
 # break  
  
 # 讓視窗保持開啟狀態，直到使用者按下 "q" 鍵或關閉視窗。  
 if cv2.waitKey(100) & 0xff == ord('q'):  
 break  
  
# 釋放資源  
cam.release()  
  
# 關閉視窗  
cv2.destroyAllWindows()