Python 小组项目报告

“形影不离”的摄像机

小组名称：爬爬小队

小组成员：辛政先、李宇轩

指导老师：鲍杨

目录

“形影不离”的摄像机

1. 简介

我们学习并使用openCV库、robomaster机器人库来实现机器人人体识别并跟随这一功能。适用于热爱拍摄户外vlog的用户，可以实现“解放双手”的效果，让摄像机自动跟随被拍摄者，并且可以随时开始或结束拍摄，以独特的角度，轻松地拍出与众不同的vlog视频。我们主要从人体识别的功能出发，先在电脑摄像头通过运用openCV库实现人体识别这一效果，并将其配置到机器人环境中，基于这一效果进一步进行研究，通过机器人库来最终实现跟随人体这一功能，并加上其他额外功能如自动避障和录制按钮。

2. 问题缘起

2.1 灵感来源

Vlog是近年来十分火热的一种拍摄视频的方式，又被称作视频博客或者视频日记。越来越多热爱拍摄视频的人选择使用Vlog来记录自己的生活，在拍摄后还可以进行剪辑美化并上传到相应视频网站与他人分享。Vlog与普通视频较大的区别在于，Vlog往往强调真实性，且主人公一般会亲自出现在镜头前。这就会出现一个问题，因为vlog大多是在移动场景下拍摄，无法持续使用三脚架等摄像机固定工具，所以往往需要拍摄者手持摄像机。需要长时间手持摄像机也是许多Vlog拍摄者的“噩梦”。因此我们小组决定编写程序来解决这个问题。

2.2 项目目标

通过编程来实现携带摄像机的机器人跟随视频拍摄者，同时增加一些额外功能提高 Vlog 拍摄效率与避免可能出现的一些问题。

3. 项目程序介绍

3.1 人体（人脸）识别部分（辛政先负责）

想要实现项目目标——机器人跟随人体，首先要实现的是机器人能够识别到人体。而我们咨询 robomaster 校队的学长后，学长推荐了 openCV 库作为视觉识别部分的编程基础。openCV 库常在机器人大赛中的视觉识别中被应用到，是一个效率较高且兼容于 robomaster 机器人的库。虽然在专业比赛中通常使用 C++语言（C++语言在机器人编程与运行中效率更高）来使用 openCV 库，但 openCV 库也提供了 python 接口来编写。于是我们决定使用 openCV 库作为人体识别功能的基础。但是在做完该部分后，发现我们使用的 robomaster S1 由于版本比学长所使用的的低，无法兼容 openCV 库，我们后来改用内置的视觉识别模块进行该部分的编程，如果版本支持 openCV 库的机器人，也可以稍加修改我们的代码，并应用到机器人上。

3.1.1openCV 库介绍

OpenCV 是一个旨在解决计算机视觉问题的 Python 库。OpenCV 最初由 Intel 在 1999 年开发，但是后来由 Willow Garage 资助。它支持很多编程语言，如 C++，Python，Java 等等。它也支持多种平台，包括 Windows，Linux 和 MacOS。

其实现人脸识别的主要原理是：通过 openCV 自带的级联分类器，可以从中提取我们所需要的特征，如人脸特征和人体特征，减少我们进行训练的步骤，再通过摄像头获取当前图片的信息，通过 NumPy 数组的形式来形成当前图片的特征文件（具有面部矩形坐标的数组），然后将特征文件与分类器中的特征文件相对比，比较其差异并确定是否是人脸。最后通过矩形框来显示图像。

3.1.2 代码结构与常见错误

主要代码结构如下：

```
import sys,os
import cv2

class recognize_face():
    def main(self):
        n = 0
        picture=cv2.VideoCapture(0,cv2.CAP_DSHOW)

        #使用人脸识别分类器
        classfier = cv2.CascadeClassifier("D:/Python/oott/Lib/site-packages/cv2/data/haarcascade_frontalface_default.xml")
        while picture.isOpened():
            #数据读取与处理
            ok, frame = picture.read()
            frame = cv2.flip(frame,1)

            #转化灰度图像
            grey = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

            #检测是否符合人脸
            face = classfier.detectMultiScale(grey, scaleFactor = 1.1, minNeighbors = 4, minSize = (50, 150))
```

```
            if len(face) > 0:
                n+=1
                for face in face:
                    x, y, w, h = face
                    cv2.rectangle(frame, (x - 10, y - 10), (x + w + 10, y + h + 10), (0, 255, 0) , 1)

                if n >= 3:
                    n = 0
                    #报错
                    os.system("sudo aplay 4611.wav")

            #显示人脸识别图像
            cv2.imshow(' ', frame)
            #Q键结束程序
            c = cv2.waitKey(10)
            if c & 0xFF == ord('q'):
                break

        #关闭摄像头与窗口
        picture.release()
        cv2.destroyAllWindows()

if __name__ == '__main__':
    recognize_face().main()
```

运行时的曾出现的报错：

第一次运行时打开摄像头的过程中报错

我们将原代码的

```
def main(self):
        n = 0
        picture = cv2.VideoCapture(0)
```

修改为

```
def main(self):
    n=0
    picture=cv2.VideoCapture(0,cv2.CAP_DSHOW)
```

修改后发现代码在启用 opencv 库中的识别分类库时无法寻找到文件

```
classfier = cv2.CascadeClassifier("haarcascade_fullbody.xml")
```

这一行原代码为写出具体路径，导致代码在不同环境下运行无法正确找到文件，于是我们增加了具体路径

```
classfier=cv2.CascadeClassifier("D:/Python/oott/Lib/site-packages/cv2/data/haarcascade_frontalface_default.xml")
```

需要注意的是，为使代码在不同机器上均能运行，需对此处路径做出相应调整，使它指向 opencv 库中的分类识别文件。

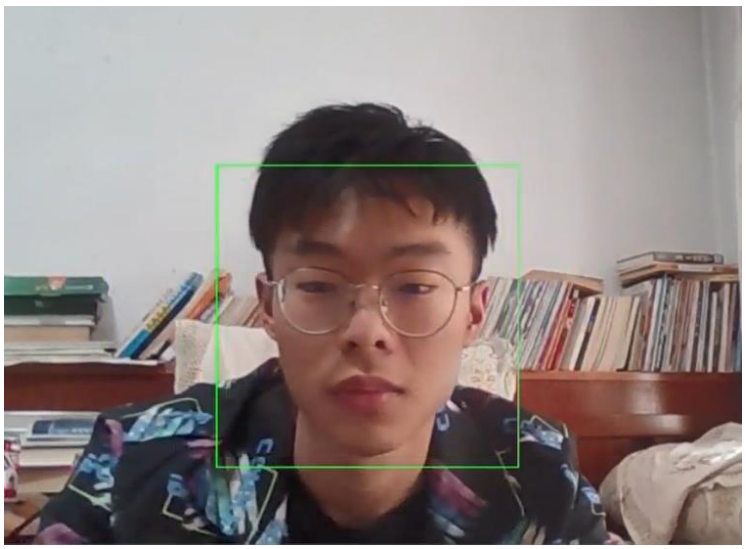
而这里，如果希望代码实现对人不同部位的识别，比如完整身体、眼睛等，可以调整此行代码对应的分类识别文件。

完整身体对应识别文件为 haarcascade_fullbody.xml

眼睛对应识别文件为 haarcascade_eye.xml

其他具体部位对应文件均可在 cv2/date 文件中找到。

效果预览：



3.2 机器人部分

3.2.1 机器人环境配置

我们所使用的的机器人是 robomaster s1（ps：任何版本高于 robomaster s1 的大疆编程机器人都可以兼容并运行我们的程序），是大疆的一款搭配摄像头及其他传感器的编程机器人，支持 scratch 和 python 两种编程语言，其中 python 仍为测试版，官方只放出部分的 python API，且没有提供相应的编程环境 SDK。我们咨询了 robomaster 校队的学长，他们使用的版本高于我们，且因为 C 语言的效率更高，他们使用的都是基于 c++的编程，故无法给我们提供相应的编程环境。于是我们查阅了大疆社区论坛、github 等，找到了以下资料。我们程序的编辑和运行需要经过以下配置：

1. 运行程序终端：

Robomaster

下载地址： https://www.dji.com/cn/robomaster-s1/downloads

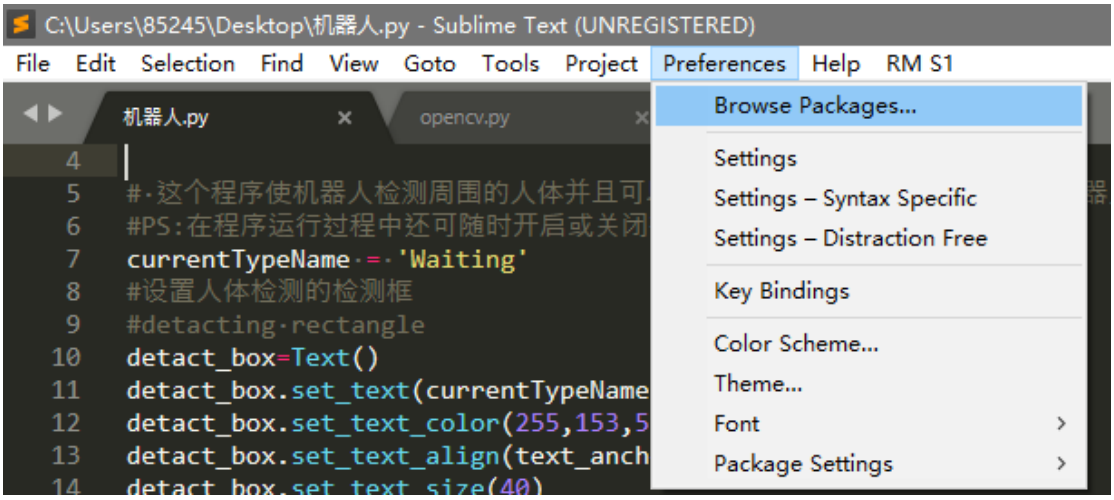
主要用于机器人的程序的最终运行和效果展示

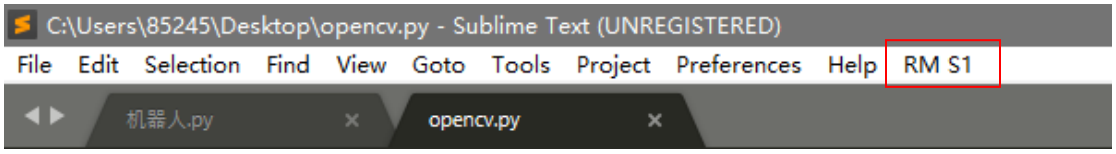
2. Python SDK 编程环境下载（在 github 上查阅到）：

下载地址：

https://github.com/open-ai-robot/awesome-dji-robomaster

注意:该 SDK 只支持 Sublime Text 3,可以通过工具栏的 Preferences-Browse Packages 进行安装，安装完后工具栏出现 RM S1 即为配置成功。





3. Python 官方 API(机器人 python 接口)以及其他 API:

官方 API:

https://www.dji.com/cn/robomaster-s1/programming-guide

更多 API(论坛中找到的):

https://robomaster-dev.readthedocs.io/zh_CN/latest/python/python_api.html

因为目前该机器人关于 python 编程仍在继续优化,后续的一些新的功能、接口或者一些相关信息可以通过查阅官方网站、大疆社区论坛和 Github 来进行了解,网址如下:

官方网站:

https://www.dji.com/cn/robomaster-s1/

大疆社区论坛:

https://bbs.dji.com/

Github:

https://github.com/

3.2.2 机器人自定义 UI 界面与人体识别 (李宇轩、辛政先共同完成)

因为 openCV 和机器人版本不兼容，我们改用了机器人内置视觉模块，为了能达到同样的效果，我们先对机器人 UI 界面进行编程，使得人体识别能更加直观的体现出来。

开始前先设置屏幕的基本参数

```
#屏幕的相关参数
#screen information
global ScreenWidth
ScreenWidth = 1920
global ScreenHeight
ScreenHeight= 1080
```

首先，在 UI 界面上添加人体识别框：

```
typeName = 'Waiting'
#设置人体检测的检测框
#detacting rectangle
detact_box=Text()
detact_box.set_text(typeName)
detact_box.set_text_color(255,153,51,200)
detact_box.set_text_align(text_anchor.middle_center)
detact_box.set_text_size(40)
detact_box.set_background_color(255,0,0,10)
detact_box.set_border_color(231,1,72,255)
detact_box.set_border_active(True)
detact_box.set_background_active(True)
detact_box.set_size(300,300)
stage.add_widget(detact_box)
```

未检测到人时

检测到人后

以上是人体检测框的具体参数

（其中人体识别框的位置由收集并处理后的人体检测数据确定）

然后用 Stage.add_widget()将此识别框加入 UI 系统

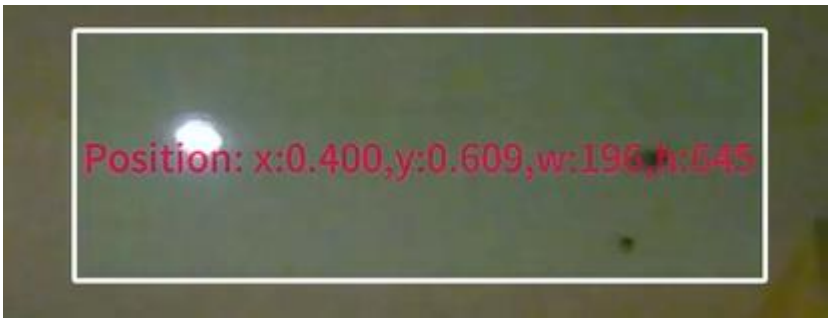
其次，在 UI 界面上添加位置信息框，来实时反馈人体的位置信息

```
#显示实时检测人体的相关参数（人的位置坐标与高度宽度）
#return·the·data·of·the·detacted·person(location,·height·and·width)
global·Postion_inf
Position_inf·=·Text()
Position_inf.set_size(540,192)
Position_inf.set_position(0,350)
Position_inf.set_text_color(231,1,72,200)
Position_inf.set_text_size(30)
Position_inf.set_border_active(True)
Position_inf.set_background_active(True)
Position_inf.set_background_color(0,255,255,12)
Position_inf.set_text('detacting......')
Position_inf.set_text_align(text_anchor.middle_center)

stage.add_widget(Postion_inf)

def·log(content):
····Position_inf.set_text('Position:·'+content)
····
```

以上是位置信息框的具体参数



实时反馈人体检测数据（x，y 坐标与 w 宽度 h 高度，均为相对数值）

其中 log（content）函数用于接收人体检测位置数据，实时更新。

然后加入检测提示框和录制按钮：

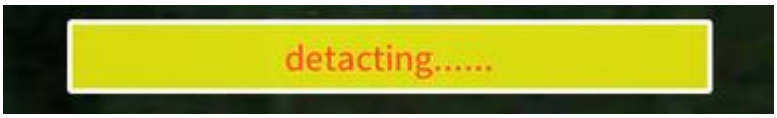
```
#检测提示框
#prompt box
global prompt_box
prompt_box = Text()
prompt_box.set_size(500,50)
prompt_box.set_border_active(True)
prompt_box.set_background_active(True)
prompt_box.set_position(0,500)
prompt_box.set_text_color(255,0,0,200)
prompt_box.set_text('detacting.....')
prompt_box.set_background_color(255,255,0,200)
prompt_box.set_text_align(text_anchor.middle_center)
stage.add_widget(prompt_box)

def prompt(inf):
    prompt_box.set_text(inf)
#录制按键
#video button
def start_video(video_button):
    media_ctrl.record(1)
    video_button.set_background_color(250,0,0,230)
    video_button.set_text('stop')
    video_button.callback_register('on_click',stop_video)
def stop_video(video_button):
    media_ctrl.record(0)
    video_button.set_background_color(0,255,0,230)
    video_button.set_text('start')
    video_button.callback_register('on_click',start_video)
video_button = Button()
video_button.set_text_color(0,0,0,200)
video_button.set_text_align(text_anchor.middle_center)
video_button.set_text_size(40)
video_button.set_text('start')
video_button.set_background_color(0,255,0,230)
video_button.set_position(-850,0)
video_button.set_active(True)
stage.add_widget(video_button)
video_button.set_size(100,100)
video_button.callback_register('on_click',start_video)
```

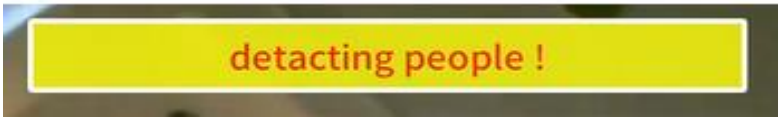
其中 prompt_box 是提示框，video_button 是录制按钮

Promot_box 未检测到人时将显示‘detacting…’检测到人时将变为

‘detacting people!’



未检测到人时



检测到人后

Video_button 可以让用户随时随地通过点击按钮来开始/结束录制

点击前

点击后

具体参数见上图

最后将收集到的人体检测数据列表提供给人体检测框函数

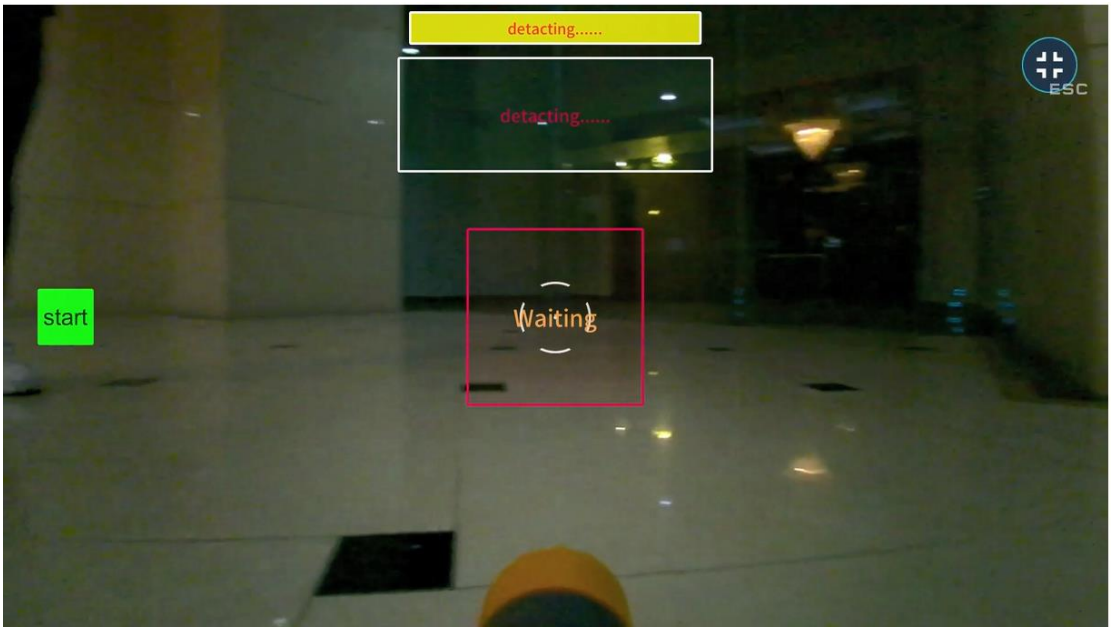
```
#人体检测数据
#the information list of the detacted person

def vision_recognized_people(msg):
    msgList = vision_ctrl.get_people_detection_info()
    if msgList[0] > 0:
        x = msgList[1]
        y = msgList[2]
        w = msgList[3]
        h = msgList[4]
        typeName = 'People'
        drawRect(x,y,w,h,typeName)
```

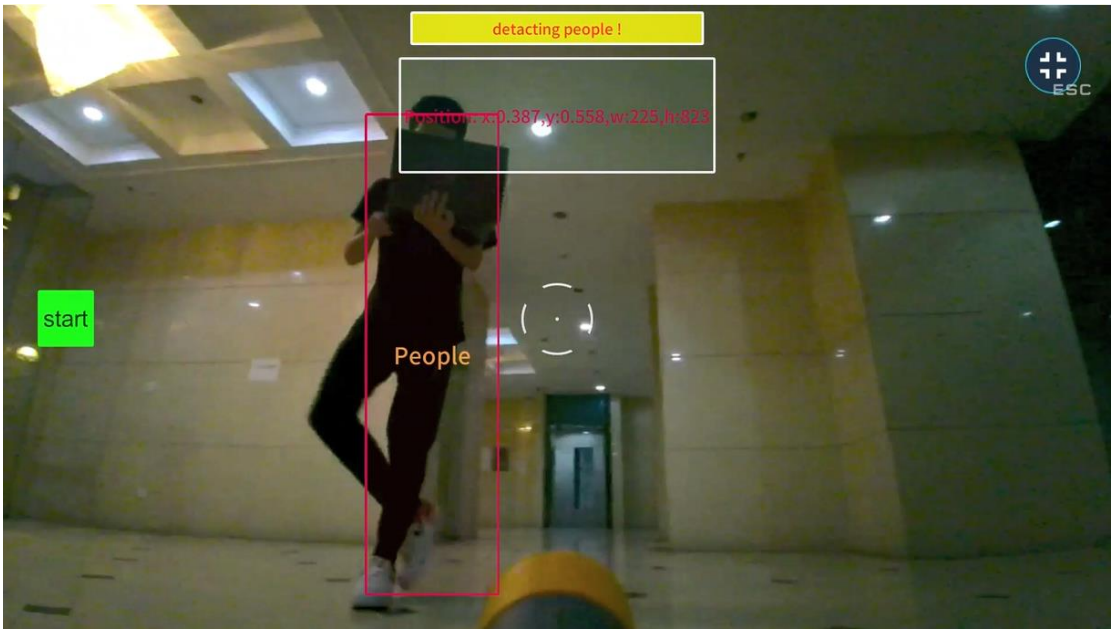
```
#人体检测框函数
#the function of detacting rectangle........
def drawRect(x,y,w,h,content):
    global ScreenWidth
    global ScreenHeight
    
    rect_x = int((x-0.5)*ScreenWidth)
    rect_y = int((0.5-y)*ScreenHeight)
    rect_w = int(w * ScreenWidth)
    rect_h = int(h*ScreenHeight)
    detact_box.set_position(rect_x,rect_y)
    detact_box.set_size(rect_w,rect_h)
    detact_box.set_text(content)
    log('x:'+str('%.3f' %x)+',y:'+str('%.3f' %y)+",w:"+str(rect_w)+',h:'+str(rect_h))
    prompt('detacting people !')
```

以实现人体检测框跟随人体的效果

最终 UI 界面效果：



检测前



检测后

3.2.3 机器人移动与跟随（李宇轩负责）

在设置完 UI 界面并且成功实现人体检测这一功能后，我们进一步研究如何使机器人摄像头跟随人体

首先我们将人体检测框的实时位置与摄像头准星的实时位置进行比较，并计算其相对差值，通过旋转摄像头使得摄像头准星对准人体检测框：

```
#行人跟随与摄像头跟随函数
#the function of human following and camera following
def follow_people():
    msgList = vision_ctrl.get_people_detection_info()
    aimList = media_ctrl.get_sight_bead_position()
    if msgList[0]>0 and aimList[0]>0:
        Yaw=96*(float(msgList[1])-float(aimList[0]))
        Pitch=54*(float(aimList[1])-float(msgList[2]-0.05))
        gimbal_ctrl.rotate_with_speed(Yaw,Pitch)
        time.sleep(0.05)
```

其中 Yaw 与 Pitch 分别为人体检测框与摄像头准星在航向轴和纵向轴的相对距离，然后调用摄像头旋转函数可以将摄像头实时转向人体检测框。

Time.sleep()是数据实时更新的间隔时间，时间越短，更新越快。

然后，在机器人距离较远的时候，我们想实现机器人跟随人体这一功能，于是我们通过人体检测框的高度来判断距离的远近，人体检测框高度越小，人体越远，此时机器人应该朝向人体运动。相关代码如下：

```
    if msgList[0]>0:
        H=int(msgList[4]*ScreenHeight)

        if H<600:
            robot_ctrl.set_mode(rm_define.robot_mode_chassis_follow)
            chassis_ctrl.move(0)
        else:
            robot_ctrl.set_mode(rm_define.robot_mode_free)
            chassis_ctrl.stop()
    else:
        gimbal_ctrl.rotate_with_speed(0,0)
```

其中 600 是临界高度，可以通过用户所需要机器人距离的远近进行调整。

最后是 start（）函数，此函数类似 main（）函数，在机器人运行时就会执行此程序。因此我们将上面的跟随函数添加进来：

```
#运行函数
#start the robot
def start():

    vision_ctrl.enable_detection(rm_define.vision_detection_people)
    media_ctrl.enable_sound_recognition(rm_define.sound_detection_applause)
    vision_ctrl.enable_detection(rm_define.vision_detection_pose)

    while vision_ctrl.check_condition(rm_define.cond_recognized_people)!=True:
        robot_ctrl.set_mode(rm_define.robot_mode_free)
        gimbal_ctrl.rotate(rm_define.gimbal_left)
    while True:
        follow_people()
        stop_robot()

    time.sleep(3600)
```

第一个 while 函数可以让机器人向四周探索直到找到人为止。

3.3 程序运行的主要步骤

下载 robomaster 程序以及配置 SDK 环境后，在 robomaster 程序的实验室中，选择 python（beta）新建程序并粘贴代码，然后连接机器人（robomaster s1/robomaster ep 及更高版本的机器人），点击右上角的运行即可。

运行界面

3.4 具体效果展示

见所附视频

4. 项目的不足与展望

因为 robomaster s1 的 python 接口仍是测试版，许多想要改进，增加的功能都无法完成，一些第三方库兼容性较低。在项目改进这一方面，我们想之后再加上声音识别功能，可以根据用户的语音来操控机器人。如果拓展接口允许，我们还想加入机械臂，通过程序控制机械臂，做出更多

不一样的效果！

5. 结论与感想

这次由于疫情原因，我们小组两名成员只能各自在家里进行编程，与其他小组不同的是，我们的机器人编程过程中，大多数函数和程序需要在机器人上运行试错,，且在查找 bug 过程中机器人往往没有返回值让我们来检查，一点点小错误就可能让机器人不运行。而我们小组采取了线上视频共同编程的方法，开摄像头，让每个人都能随时运行机器人来测试程序，因祸得福，也提高了我们小组成员间的合作能力。因为该机器人的 python 接口仍为测试版，网上的现成资料较少。所有我们花费了很多时间在查阅资料与学习的过程中，但是当我们做出程序时，感到十分自豪与满足，这次编程经历，我们都从中收获了许多！