利用 AirSim 图像 api 连接 Yolov5 实现实时目标检测

通过 AirSim 的封装方法 simGetImage 获取图像，将图像转换为 OpenCV 格式后进行检测(将二进制图像数据解码为 OpenCV 格式的 BGR 图像)，对 AirSim 的配置文件进行更改，调整"CameraDefaults"其 api 中的内容来改变获得的数据，在通过使用 draw 函数在图像上绘制检测框，并打印检测到的类别名称，并显示处理后的图像。功能实现的代码分为两部分，其一是用于从 AirSim 模拟环境中实时获取图像(airsim_detect.py)，另一个是用于从视频文件中逐帧读取图像，并进行物体检测和标注(yolo_orin_api.py)。

以下列出所需修改的 AirSim 配置文档 settings.json 文件：

```
{
  "SeeDocsAt": "https://github.com/Microsoft/AirSim/blob/master/docs/settings.md",
  "SettingsVersion": 1.2,
  "SimMode": "Multirotor",
  "CameraDefaults": {
    "CaptureSettings": [
      {
        "ImageType": 0,
        "Width": 1024,
        "Height": 768,
        "FOV_Degrees": 90,
        "AutoExposureSpeed": 100,
        "AutoExposureBias": 0,
        "AutoExposureMaxBrightness": 0.64,
        "AutoExposureMinBrightness": 0.03,
        "MotionBlurAmount": 0,
        "TargetGamma": 1.0,
        "ProjectionMode": "",
        "OrthoWidth": 5.12
      }
    ]
  },
    "Vehicles": {
      "SimpleFlight": {
        "VehicleType": "SimpleFlight",
        "RC": {
          "RemoteControlID": 0
        }
      }
    }
  }
```

其中 CameraDefaults 就是我们添加的摄像头信息，其内部的项目都是对应的英文名意思，不做过多赘述，只需注意设置分辨率的部分，此文件内分辨率是 1024*768，会决定弹出的窗口清晰度与大小。

以下是 airsim_detect.py 文件代码：

```
import time
import airsim
import numpy as np
import yolo_orin_api
import cv2
client = airsim.MultirotorClient(ip="192.168.123.102")
camera_name = '0'   # 前向中间   0,底部中间 3
image_type = airsim.ImageType.Scene   # 彩色图 airsim.ImageType.Scene, Infrared
while 1:
    response = client.simGetImage(camera_name, image_type, vehicle_name='')   # simGetImage 接口的调用方式如下
    if response:
        # 转换成 opencv 图片格式
        img_bgr = cv2.imdecode(np.array(bytearray(response), dtype='uint8'), cv2.IMREAD_UNCHANGED)   # 从二进制图片数据中读
        # boxes 格式为[[left, top, right, bottom], ...]
        boxes, classes, scores = yolo_orin_api.yolo_recognition(img_bgr)
        if boxes is not None:
            yolo_orin_api.draw(img_bgr, boxes, scores, classes)
            index = 0
            for cat in classes:
                cat_name = yolo_orin_api.CLASSES[cat.astype(int)]
                print(cat_name)

        # show output
        cv2.imshow("preview", img_bgr)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    print("--------------------")
    time.sleep(0.1)
```

这段代码实现了通过 AirSim 获取无人机摄像头图像，并使用 YOLO 目标识别算法对图像进行处理，最后在 OpenCV 中显示识别结果。以下是代码的主要步骤和功能描述：

导入所需的库：导入了时间处理模块 time，AirSim 模拟器的客户端库 airsim，数组处理库 numpy，YOLO 目标识别算法的 API yolo_orin_api，以及图像处理库 cv2（OpenCV）

初始化 AirSim 客户端：创建了一个 MultirotorClient 对象，并指定了连接的 IP 地址为 "192.168.123.102"

设置摄像头参数：指定了摄像头名称 camera_name 为 '0'（前向中间摄像头），并选择了图像类型 image_type 为 airsim.ImageType.Scene（彩色图像）。

进入循环：使用 while 1 进入一个无限循环，即持续地获取和处理图像数据。

获取图像数据：通过 client.simGetImage() 方法获取摄像头图像的二进制数据，并转换成 OpenCV 图像格式 img_bgr

使用 YOLO 进行目标识别：调用 yolo_orin_api.yolo_recognition() 方法对图像进行目标识别，获取识别到的物体边界框 boxes 类别 classes 和置信度 scores。

绘制识别结果：使用 yolo_orin_api.draw() 方法在图像上绘制识别结果，将识别到的物体用边界框标记出来。

输出识别结果：遍历识别到的每个物体类别，打印类别名称，并将识别结果显示在图像窗口中。

监听按键事件：使用 cv2.waitKey() 监听键盘事件，当按下 'q' 键时退出循环。

延时处理：在每次循环结束后，通过 time.sleep(0.1) 延时 0.1 秒，以控制循环的执行速度。

这段代码的主要作用是实时从 AirSim 中获取图像并进行目标识别，然后在 OpenCV 窗口中显示识别结果，用户可以通过按下 'q' 键来停止程序运行。直接粘贴代码格式或许会有问题，为求准确可使用原编码文件。

以下是 yolo_orin_api.py 文件代码：

```
import cv2
import torch
import torch.hub
import time
CLASSES = ("drone", "car", "person", "bird", "cat", "dog")
# Model
model_path = r'D:\yolov5-master\Airsim connect\my_best.pt'
model = torch.hub.load('ultralytics/yolov5', 'custom', path=model_path)
def draw(image, boxes, scores, classes):
    for box, score, cl in zip(boxes, scores, classes):
        left, top, right, bottom = box
        left = int(left)
        top = int(top)
        right = int(right)
        bottom = int(bottom)
        cl = int(cl)
        # 修改框的颜色为红色 (0, 0, 255)
        cv2.rectangle(image, (left, top), (right, bottom), (0, 0, 255), 2)
        # 定义标签文本
        label = '{0} {1:.2f}'.format(CLASSES[cl], score)
        label_size, base_line = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.6, 1)
        # 添加红色底色的标签
        cv2.rectangle(image, (left, top - label_size[1] - base_line),
                      (left + label_size[0], top), (0, 0, 255), cv2.FILLED)
        # 将文字颜色修改为白色
        cv2.putText(image, label, (left, top - base_line),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2,
lineType=cv2.LINE_AA)
def yolo_recognition(img):
    """
    yolo 识别物体
    :param img:
    :return boxes, classes, scores: boxes 格式为[[left, top, right, bottom], ...]
    """
    results = model(img)
    # xmin, ymin, xmax, ymax, confidence, class
    rec_result = results.xyxy[0]
    rec_result_np = rec_result.cpu().numpy()
    boxes = rec_result_np[:, 0:4]
    score = rec_result_np[:, 4]
    classes = rec_result_np[:, 5]
    return boxes, classes, score
if __name__ == "__main__":
    # videos
    video_path = "sample_720p.mp4"
    cap = cv2.VideoCapture(video_path)
    frames, loopTime = 0, time.time()
    while cap.isOpened():
        frames += 1
        ret, img = cap.read()
        # Inference
        boxes, classes, scores = yolo_recognition(img)
        if boxes is not None:
            draw(img, boxes, scores, classes)
        # show output
```

```
            cv2.imshow("post process result", img)
            # Press Q on keyboard to exit
            key = cv2.waitKey(1)  # 等待按键命令, 1000ms 后自动关闭
    # Closes all the frames
    cap.release()
    cv2.destroyAllWindows()
```

这段代码实现了使用 YOLOv5 模型对视频中的物体进行实时识别和标注，并通过 OpenCV 在窗口中显示识别结果。以下是代码的主要步骤和功能描述：

导入所需的库：导入了图像处理库 cv2，深度学习框架 PyTorch 和 torch.hub，以及时间处理模块 time

定义物体类别：通过 CLASSES 列表定义了待识别的物体类别，包括"drone"、"car"、"person"、"bird"、"cat" 和 "dog"

加载模型：使用 torch.hub.load() 方法加载 YOLOv5 模型，指定模型的路径为 model_path。

定义绘制函数：定义了 draw() 函数，用于在图像上绘制识别结果的边界框和标签，并调整了标签文本的颜色和样式

定义识别函数：定义了 yolo_recognition() 函数，用于调用 YOLOv5 模型对图像进行目标识别，返回识别结果的边界框、类别和置信度

主程序入口：在 __main__ 部分，设置了视频路径 video_path 并通过 OpenCV 打开视频文件。

处理视频帧：通过 cv2.VideoCapture() 方法打开视频文件，然后进入一个循环，每次读取视频的一帧图像，并调用 yolo_recognition() 函数进行目标识别。

绘制识别结果：如果成功识别到物体，则调用 draw() 函数在图像上绘制识别结果的边界框和标签

显示识别结果：使用 cv2.imshow() 方法在窗口中显示处理后的图像，并通过 cv2.waitKey() 监听键盘事件，按下 'q' 键时退出循环

清理资源：在循环结束后释放视频流资源，并关闭所有窗口

这段代码的主要功能是使用预训练的 YOLOv5 模型实现对视频中物体的实时识别和标注，可以通过修改 CLASSES 列表和视频路径来适应不同的物体识别任务和视频数据源，直接粘贴代码格式或许会有问题，为求准确可使用原编码文件。

文件修改完成后，先启动 AirSim(即进入 UE 场景点击 Play),而后在终端运行：
python airsim_detect.py
则会出现 preview 窗口显示识别情况，终端会实时返回识别信息，按 Q 即可关闭窗口。

```
(yolov5) PS D:\yolov5-master\Airsim connect> python .\airsim_detect.py
Using cache found in C:\Users\Administrator/.cache\torch\hub\ultralytics_yolov5_master
YOLOv5  2024-6-3 Python-3.8.19 torch-2.0.1+cu118 CUDA:0 (NVIDIA GeForce GTX 1650, 4096MiB)

Fusing layers...
YOLOv5s_drone summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
Adding AutoShape...
drone
drone
drone
--------------------
drone
drone
drone
--------------------
drone
drone
drone
--------------------
```