1.实验名称及目的

1.1实验名称

无人机跟随圆形案板移动实验

1.2实验目的

本实验旨在通过控制无人机跟踪一个移动的圆形目标来演示基于计算机视觉的无人机自主跟随技术。通过该实验，学习者可以掌握以下技能：

1. 理解基于视觉的目标检测与跟踪原理；
2. 掌握利用图像处理算法识别特定颜色目标的方法；
3. 学习如何根据目标在图像中的位置和大小变化计算控制指令；
4. 实践无人机的视觉伺服控制，实现对移动目标的实时跟随；
5. 熟悉RflySim仿真平台中视觉系统与控制系统的集成应用；
6. 了解PID控制在无人机视觉跟随中的实际应用。

通过手动控制和自动控制两种模式，学习者可以深入理解视觉伺服控制的基本原理和实现方法，为进一步研究复杂场景下的目标跟踪打下基础。

1.3关键知识点

整体思路和框架

本实验的核心是实现一个基于视觉的无人机跟随控制系统，系统主要由以下几个模块组成：

1. **视觉采集模块**：通过RflySim的视觉系统获取无人机摄像头拍摄的画面；
2. **图像处理模块**：对采集到的图像进行处理，识别其中的红色圆形目标；
3. **控制算法模块**：根据目标在图像中的位置和大小，计算无人机的期望速度；
4. **无人机控制模块**：将控制指令发送给无人机，使其能够跟随目标移动。

整个系统采用多线程方式运行，其中一个线程负责图像采集和处理，另一个线程负责控制指令计算和发送。

程序解析

键盘控制程序(circle_follow_kbCtrl.py)

键盘控制程序允许用户通过键盘控制圆形目标的移动，同时无人机自动跟随该目标。程序主要包含以下几个关键部分：

1. **类初始化和参数设置**：

```python
class CircleFollower:
    """
    无人机视觉跟随圆形目标控制类
    """
    def __init__(self):
        # 初始化 API 对象
        self.ue = UE4CtrlAPI.UE4CtrlAPI()
        self.vis = VisionCaptureApi.VisionCaptureApi()
        self.mav = PX4MavCtrl.PX4MavCtrler()

        # 运行状态
        self.running = False
        self.detect_flag = False

        # 目标位置 [x, y, z, yaw]
        # 注意: UE4坐标系可能与直觉不同，这里保留原代码的初始值
        self.circle_pose = [2.0, 0.0, -2.0, 0.0]

        # 图像处理参数
        self.img_center_x = 320
        self.img_center_y = 240
        self.tolerance_center = 15  # 中心偏差容忍度
        self.tolerance_dist = 15    # 距离(大小)偏差容忍度

        # PID 控制参数 (比例系数)
        self.k_x = 0.015  # 前后速度系数 (基于目标大小变化)
        self.k_y = 0.009  # 左右速度系数 (基于水平位置偏差)
        self.k_z = 0.005  # 上下速度系数 (基于垂直位置偏差)

        # 初始目标大小记录 (用于保持距离)
        self.init_dist_x = 0
        self.init_dist_y = 0

        # 红色阈值 (HSV)
        self.low_hsv1 = np.array([0, 43, 46])
        self.high_hsv1 = np.array([10, 255, 255])
        self.low_hsv2 = np.array([156, 43, 46])
        self.high_hsv2 = np.array([180, 255, 255])
```

2. **环境初始化函数**：

```python
def setup_environment(self):
    """初始化 UE4 环境和视觉接口"""
    print("正在初始化环境...")
    # 设置 UE4 帧率
    self.ue.sendUE4Cmd(b"t.MaxFPS 30", 0)

    # 创建圆形目标 (Vehicle 3, Type 809)
    self.ue.sendUE4PosNew(3, vehicleType=809, PosE=self.circle_pose[:3])

    # 加载视觉配置
    self.vis.jsonLoad()
    time.sleep(1)

    # 设置额外的动作参数 (保留原代码逻辑)
    self.ue.sendUE4ExtAct(3, ActExt=[1, 0.8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,

    # 请求图像数据
    is_success = self.vis.sendReqToUE4()
    if not is_success:
        print("错误: 请求图像失败，程序退出。")
        sys.exit(0)

    self.vis.startImgCap()
    print("视觉捕获已启动。")
    time.sleep(2)
```

3. **目标检测函数**：

```python
def detect_circle(self, img: np.ndarray) -> Tuple[List[int], List[int], List[int
    """
    图像处理：检测红色圆形
    返回: ([x_min, x_max], [y_min, y_max], [center_x, center_y])
    """
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # 红色在 HSV 空间有两个区间
    mask1 = cv2.inRange(hsv, self.low_hsv1, self.high_hsv1)
    mask2 = cv2.inRange(hsv, self.low_hsv2, self.high_hsv2)
    mask = mask1 + mask2

    nonzero = np.nonzero(mask)
    if np.size(nonzero) == 0:
        return [-1, -1], [-1, -1], [-1, -1]

    row_min = np.min(nonzero[0])
    row_max = np.max(nonzero[0])
    col_min = np.min(nonzero[1])
    col_max = np.max(nonzero[1])

    mid_row = int((row_max + row_min) / 2)
    mid_col = int((col_max + col_min) / 2)

    # 绘制十字准星用于调试
    cv2.line(img, (col_min, mid_row), (col_max, mid_row), (255, 255, 255), 1)
    cv2.line(img, (mid_col, row_min), (mid_col, row_max), (255, 255, 255), 1)

    cv2.imshow("Drone View", img)
    cv2.waitKey(1)

    return [col_min, col_max], [row_min, row_max], [mid_col, mid_row]
```

4. **控制循环函数：**

```python
def control_loop(self):
    """
    主控制循环：获取图像 -> 识别 -> 控制无人机
    """
    print("开始控制循环...")
    self.running = True

    while self.running:
        if self.vis.hasData[0]:
            img = self.vis.Img[0]
            self.vis.hasData[0] = False

            # 1. 视觉识别
            x_range, y_range, center = self.detect_circle(img)
            x_min, x_max = x_range
            y_min, y_max = y_range
            cnt_x, cnt_y = center

            # 未检测到目标
            if x_min == -1:
                time.sleep(0.001)
                continue

            if not self.detect_flag:
                self.detect_flag = True
                print("目标已锁定！")

            dist_x = x_max - x_min
            dist_y = y_max - y_min

            # 记录初始距离（作为参考大小）
            if self.init_dist_x == 0:
                self.init_dist_x = dist_x
            if self.init_dist_y == 0:
                self.init_dist_y = dist_y

            # 2. 计算控制量
            vx = 0
            vy = 0
            vz = 0

            # 横向控制（左右）
            if abs(cnt_x - self.img_center_x) > self.tolerance_center:
                vy = self.k_y * (cnt_x - self.img_center_x)

            # 垂直控制（上下）
            if abs(cnt_y - self.img_center_y) > self.tolerance_center:
                vz = self.k_z * (cnt_y - self.img_center_y)

            # 纵向控制（前后） - 基于目标在图像中的大小(高度)变化
            # 如果目标变小(dist_y < init)，说明远了，需要向前飞(vx > 0)
            if abs(dist_y - self.init_dist_y) > self.tolerance_dist:
```

```
                vx = self.k_x * (self.init_dist_y - dist_y)

            # 3. 发送指令 (前-右-下 坐标系)
            self.mav.SendVelFRD(vx, vy, vz, 0)

        time.sleep(0.001)
```

5. **键盘事件处理**：

```python
def move_target(self, direction: str):
    """
    控制圆形目标移动的回调函数
    """
    offset = 0.01  # 每次移动的距离

    if direction == "right":
        self.circle_pose[1] += offset
    elif direction == "left":
        self.circle_pose[1] -= offset
    elif direction == "up":
        self.circle_pose[2] -= offset # 注意坐标系方向
    elif direction == "down":
        self.circle_pose[2] += offset
    elif direction == "front":
        self.circle_pose[0] += offset
    elif direction == "back":
        self.circle_pose[0] -= offset

    # 更新 UE4 中物体位置
    self.ue.sendUE4PosNew(
        3,
        vehicleType=809,
        PosE=self.circle_pose[:3],
        AngEuler=[0, 0, self.circle_pose[3]],
    )
```

自动控制程序(circle_follow_autoCtrl.py)

自动控制程序实现了圆形目标的自动移动，使无人机能够在无需人工干预的情况下进行跟随测试。关键差异在于目标移动方式：

```python
def auto_move_target(self):
    """
    自动控制圆形目标:
    1. 沿着X轴一直向后(远离)移动
    2. Y轴(左右)和Z轴(上下)做正弦波运动
    3. 速度较慢，以便无人机跟随
    """
    print("启动目标自动移动 (X轴后退 + 正弦波)...")
    start_time = time.time()
    last_time = start_time

    # 记录初始位置
    current_x = self.circle_pose[0]
    center_y = self.circle_pose[1]
    center_z = self.circle_pose[2]

    # 运动参数配置
    move_speed_x = 0.3   # X轴移动速度 (米/秒) - 较慢

    amp_y = 1.0          # Y轴(左右)移动幅度 (米)
    freq_y = 0.1         # Y轴移动频率 (Hz)

    amp_z = 0.5          # Z轴(上下)移动幅度 (米)
    freq_z = 0.1         # Z轴移动频率 (Hz)

    while self.running:
        current_time = time.time()
        dt = current_time - last_time
        last_time = current_time
        elapsed_time = current_time - start_time

        # 1. X轴: 持续向后移动 (远离无人机)
        # 注意: 假设无人机在原点，目标在X正方向，向后移动即X增加
        current_x += move_speed_x * dt
        self.circle_pose[0] = current_x

        # 2. Y轴: 正弦波运动
        self.circle_pose[1] = center_y + amp_y * np.sin(2 * np.pi * freq_y * elapsed_

        # 3. Z轴: 正弦波运动 (可选，形成螺旋或波浪效果)
        self.circle_pose[2] = center_z + amp_z * np.sin(2 * np.pi * freq_z * elapsed_

        # 更新 UE4 中物体位置
        self.ue.sendUE4PosNew(
            3,
            vehicleType=809,
            PosE=self.circle_pose[:3],
            AngEuler=[0, 0, self.circle_pose[3]],
        )

        time.sleep(0.02) # 50Hz 刷新率
```

关键技术点总结

1. **视觉识别**：使用OpenCV库对图像进行HSV色彩空间转换，并通过设定红色阈值范围识别红色圆形目标；
2. **控制策略**：采用简单的比例控制(P控制)根据目标在图像中的位置和大小调整无人机速度；
3. **坐标变换**：将图像坐标系中的偏差转换为无人机机体坐标系中的速度指令；
4. **多线程处理**：使用多线程分别处理图像采集和控制指令发送，提高系统响应速度；
5. **仿真接口**：通过RflySim提供的API与仿真环境交互，控制无人机和目标物体。

2.实验效果

本实验展示了基于视觉的无人机目标跟随功能，包括手动控制和自动控制两种模式。

在手动控制模式下，用户可以通过键盘控制红色圆形目标在三维空间中的移动，而无人机能够实时跟随目标的运动。当目标在视野范围内移动时，无人机会相应地调整自身位置以保持相对距离和视角。

在自动控制模式下，红色圆形目标会按照预设轨迹（沿X轴向后移动，同时在Y轴和Z轴做正弦波运动）自动移动，无人机同样能够稳定地跟随这一复杂轨迹运动。

实验成功验证了以下效果：

1. 无人机能够准确识别并锁定红色圆形目标；
2. 当目标发生位移时，无人机能够快速响应并调整飞行姿态；
3. 在目标做复杂轨迹运动时，无人机能够平稳跟随，表现出良好的控制性能；

通过本实验，可以直观地观察到计算机视觉技术在无人机自主导航中的应用效果，为后续更复杂的视觉导航任务奠定基础。

3.文件目录

例程目录：
[安装目录]\RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e6_Circle-follow

4.运行环境

4.1 软件要求

Windows 10及以上版本；RflySim工具链。

> ①：推荐配置请见： https://rflysim.com/

4.2 硬件要求

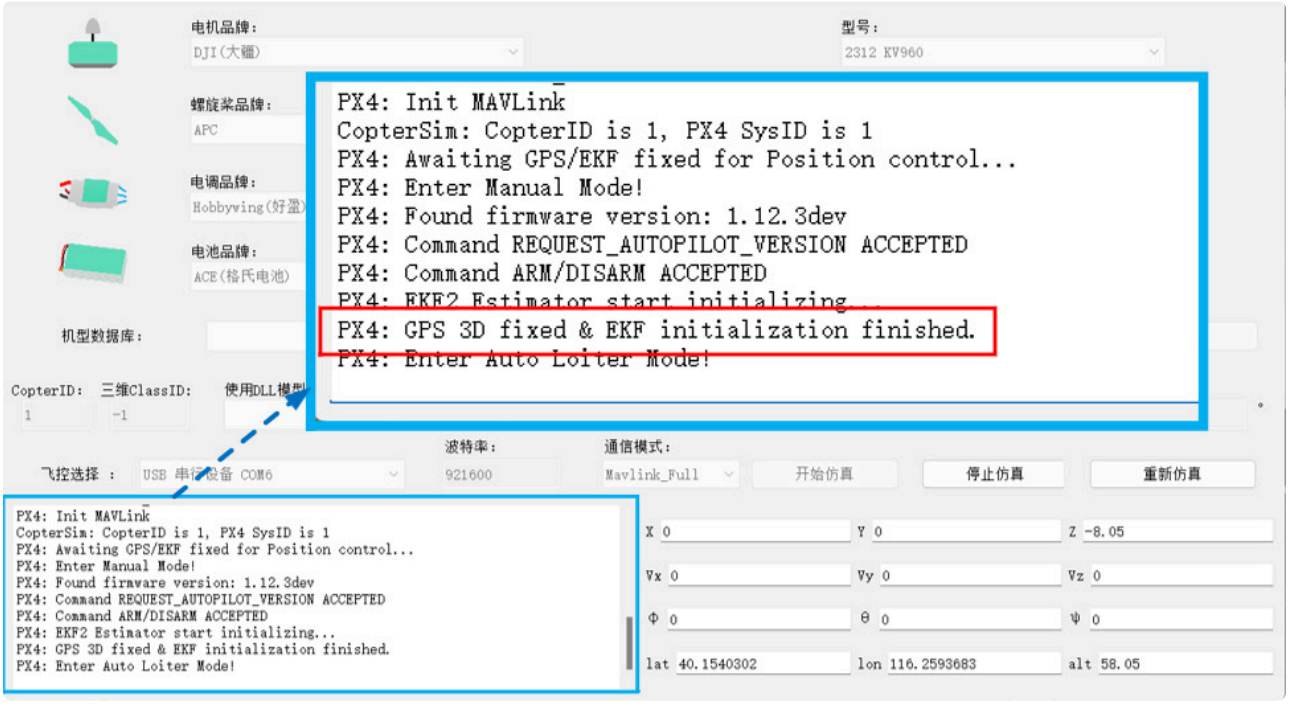
笔记本/台式电脑1台

> ①：推荐配置请见： https://rflysim.com/doc/zh/HowToInstall.pdf

5.实验步骤

5.1 启动仿真

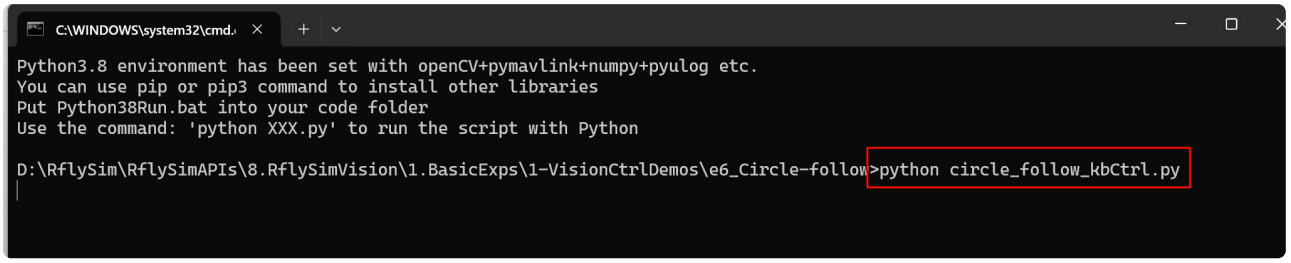
双击运行[circle_follow.bat]文件，等待仿真环境初始化完成。脚本将会启动 1 个 QGC 地面站， 1 个 CopterSim 、1 个 RflySim3D 软件， 等待CopterSim软件下侧日志栏必须打印出 `GPS 3D fixed & EKF initialization finished` 字样代表初始化完成。如下图所示：



5.2 运行键盘控制圆形案板程序

双击运行[Python38Run.bat]脚本，在弹出的窗口中输入：

```
python circle_follow_kbCtrl.py
```

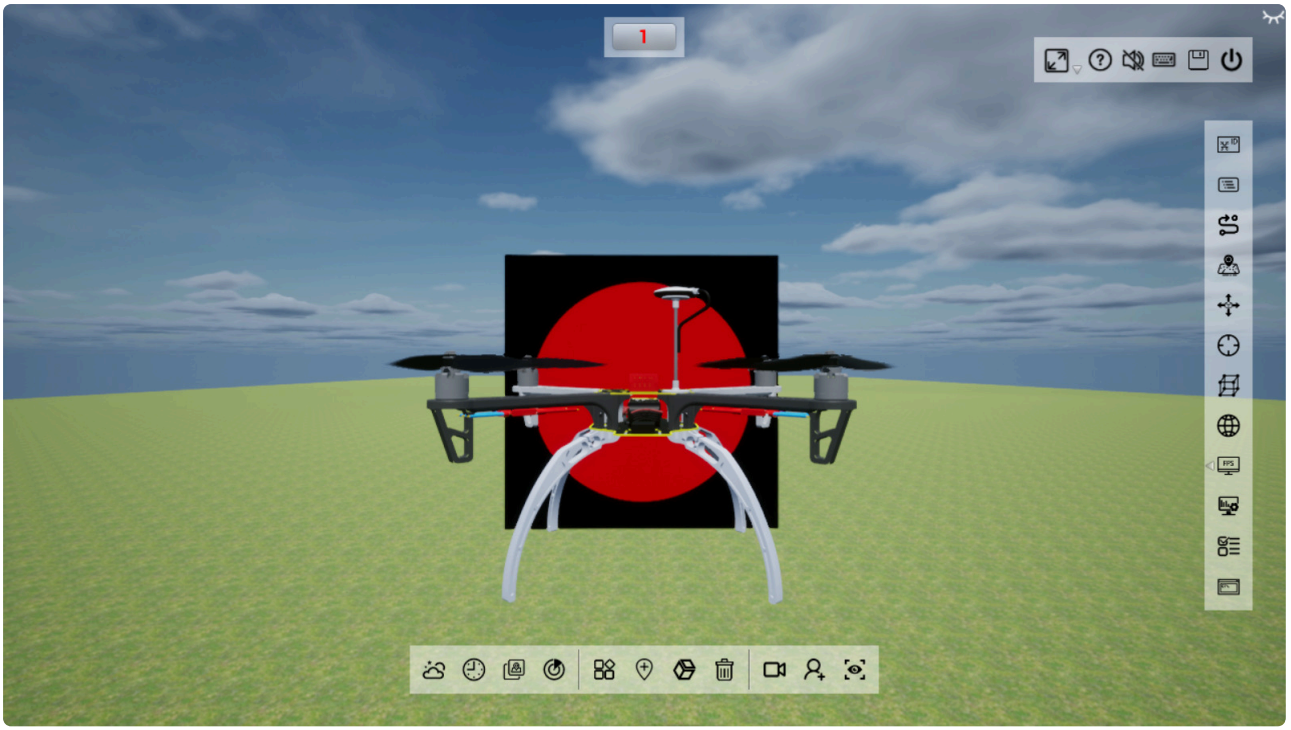


然后，等待出现

```
系统运行中...
使用方向键移动目标 (Ctrl+上下 前后移动)
按 'Esc' 键退出程序
目标已锁定！
```

字样后，使用键盘上：

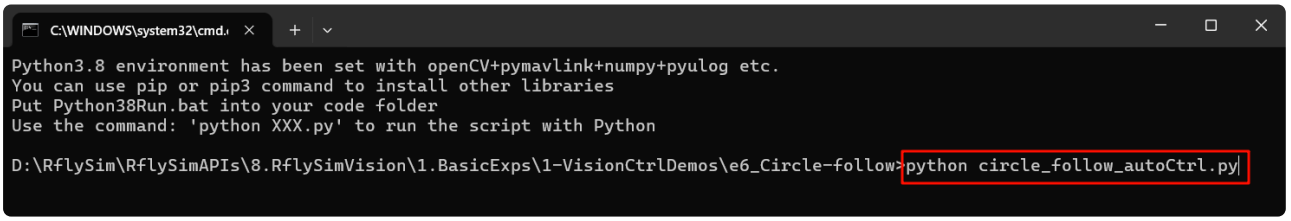
- Up(↑)：表示案板向上移动；
- Down(↓)：表示案板向下移动；
- Left(←)：表示案板向左右移动；
- Right(→)：表示案板向右移动；
- Ctrl+Up: 表示案板向飞机前方移动，(远离飞机)；
- Ctrl+Down:表示案板向飞机后移动，(靠近飞机)；
  无人机将跟随移动，如下图所示。



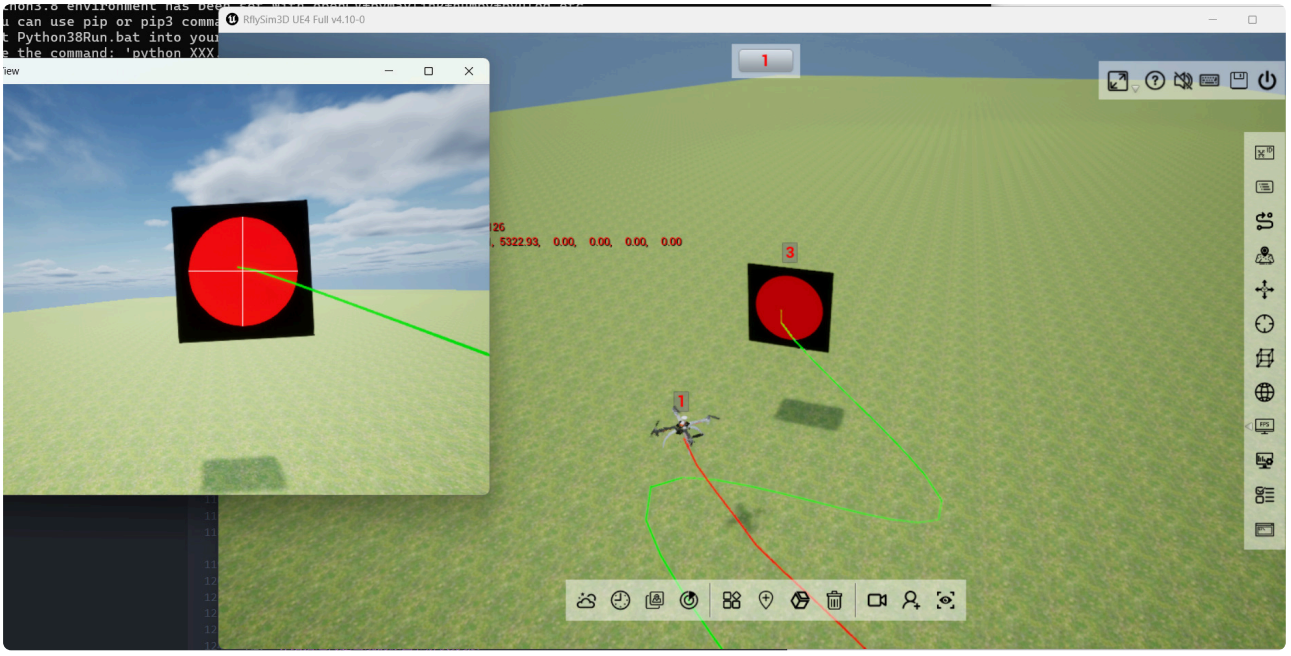
5.3 运行自动控制圆形案板程序（选做）

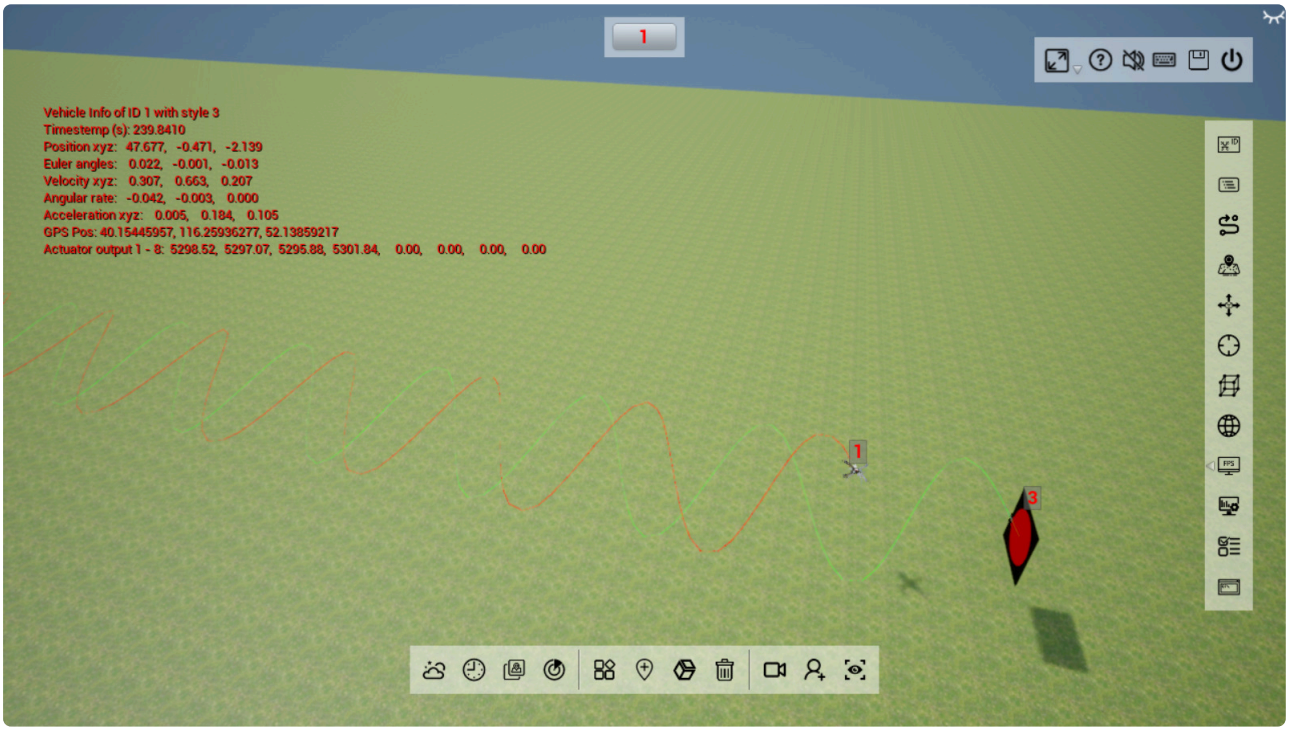
重复5.1中的步骤，确保初始化完成。双击运行[Python38Run.bat]脚本，在弹出的窗口中输入：

```
python circle_follow_autoCtrl.py
```



在RflySim3D软件中，按下“T”键即可打开载具运动轨迹显示，如下图所示，圆形案板将沿着X轴做正弦运动基于屏幕向后移动，无人机将自动跟随。





6.参考资料

1. 《OpenCV官方文档》
2. 《RflySim用户手册》
3. 《Computer Vision: Algorithms and Applications》by Richard Szeliski
4. 《Python Robotics Projects》by Dr. Lentin Joseph
5. 《Modern Control Systems》by Richard C. Dorf and Robert H. Bishop
   《Learning OpenCV 4 Computer Vision with Python》by Joseph Howse and Joe M
6. inichino

7.常见问题

Q1：仿真启动后无人机无法起飞或者飞行不稳定

A1：检查是否正确完成了初始化过程，确保CopterSim软件下侧日志栏打印出`GPS 3D fixed & EKF initialization finished`字样后再进行后续操作。如果仍然存在问题，尝试重启仿真环境并检查网络连接。

Q2：无法检测到红色圆形目标或者检测不准确

A2：这个问题通常由光照条件或颜色阈值设置引起。确保环境中红色目标足够明显且没有其他强红光干扰。可以适当调整HSV阈值参数：

```
# 红色阈值 (HSV)
self.low_hsv1 = np.array([0, 43, 46])
self.high_hsv1 = np.array([10, 255, 255])
self.low_hsv2 = np.array([156, 43, 46])
self.high_hsv2 = np.array([180, 255, 255])
```

根据实际环境光照情况微调这些参数。

Q3：无人机跟随过程中出现震荡或者响应迟缓

A3：这是控制参数不合适导致的问题。可以调整以下PID控制参数来优化跟随效果：

```
# PID 控制参数 (比例系数)
self.k_x = 0.015  # 前后速度系数 (基于目标大小变化)
self.k_y = 0.009  # 左右速度系数 (基于水平位置偏差)
self.k_z = 0.005  # 上下速度系数 (基于垂直位置偏差)
```

增大参数会使响应更快但可能导致震荡，减小参数会使响应更平滑但可能产生滞后。

Q4：程序运行时报错找不到相关模块

A4：确保在正确的RflySim Python环境中运行程序。推荐使用Python38Run.bat脚本启动Python环境，该环境已经预装了所有必需的依赖库。如果仍有问题，可以手动安装缺失的库：

```
pip install opencv-python
pip install keyboard
pip install numpy
```

Q5：键盘控制无响应

A5：某些情况下键盘事件监听可能会受到操作系统或其他程序的影响。确保以管理员权限运行程序，并关闭可能干扰键盘输入的其他应用程序。如果问题依然存在，可以尝试重新插拔键盘或更换USB接口。