AirSim的python控制接口介绍

1. 连接到AirSim：

（1）MultirotorClient类：

class MultirotorClient(VehicleClient, object):  
 def \_\_init\_\_(self, ip = "", port = 41451, timeout\_value = 3600):  
 super(MultirotorClient, self).\_\_init\_\_(ip, port, timeout\_value)

功能：连接到 AirSim 模拟器实例。

（2）MultirotorClient 类的方法confirmConnection(self)

功能：检验是否连接到 AirSim 模拟器。

参数：无。

输出：确认连接成功将会打印"Connected!"，确认连接失败会输出错误信息。

**样例**：

import airsim

# Connect to the AirSim simulator

client = airsim.MultirotorClient()

client.confirmConnection()

1. 基本配置指令

（1）Vector3r类

功能：设置三维坐标格式。

参数：x、y、z分别表示三维坐标的三个分量

样例：loc = airsim.Vector3r(0, 0, 1) #设置三维坐标(0,0,1)在airsim中的格式为loc

（2）Pose类

功能：表示环境中的物体的位置和姿态。具体来说，Pose 类包含了位置（Position）和方向（Orientation）的信息，这些信息是基于四元数（Quaternion）和三维向量（Vector3r）来描述的。

参数：position (Vector3r): 表示物体在世界坐标系中的位置。Vector3r 类包含三个属性 x\_val, y\_val, 和 z\_val 分别代表 x, y, z 轴上的坐标值；

orientation (Quaternionr): 表示物体的方向。Quaternionr 类包含四个属性 w\_val, x\_val, y\_val, 和 z\_val，它们共同定义了一个四元数，用于描述旋转。

**样例**：

position = airsim.Vector3r(10, 20, -5) # 定义位置

orientation = airsim.to\_quaternion(0.707, 0, 0.707) # 定义方向，这里是一个45度角绕Z轴旋转的例子

pose = airsim.Pose(position, orientation)

1. VehicleClient 类的方法simAddVehicle (self, vehicle\_name, vehicle\_type, pose, pawn\_path = "")

功能：创建一个新的交通工具（无人机、无人车等）

参数：vehicle\_name (str)：交通工具的名称；

vehicle\_type (str)：交通工具的类型，例如”simpleflight”；

pose (Pose)：交通工具的初始姿态；

pawn\_path (str, optional)：自定义交通工具的蓝图（在UE中的外观）。

**样例**：

import airsim

# Connect to the AirSim simulator

client = airsim.MultirotorClient()

client.confirmConnection()

position = airsim.Vector3r(10, 20, -5) # 定义位置

orientation = airsim.to\_quaternion(0.707, 0, 0.707) # 定义方向，这里是一个45度角绕Z轴旋转的例子

pose = airsim.Pose(position, orientation)

client.simAddVehicle(vehicle\_name=“drone”,vehicle\_type= “simplefight”, pose=pose)#创建一个名字叫做drone、类型为simplefight、姿态为pose的无人机

（4）VehicleClient类的方法enableApiControl(self, is\_enabled, vehicle\_name = '')。

功能：启用或关闭控制交通工具（无人机、无人车等）的控制接口。

参数：is\_enabled(bool)：True（启用），False（关闭）；

vehicle\_name：所要控制的交通工具（无人机、无人车等）的名称。

（5）VehicleClient类的方法armDisarm(self, arm, vehicle\_name = '')

功能：武装或解除武装，使控制交通工具能够移动或无法移动

参数：arm(bool)：True（武装，准备启动）、False（解除武装，关闭电机）；

vehicle\_name：所要控制的交通工具（无人机、无人车等）的名称。

（6）VehicleClient类的方法addLandingPort(self, port\_name, location, box\_num)。

功能：添加一个机场。

参数：port\_name：要添加的机场名称；

location(Vector3r)：要添加的机场的位置；

box\_num：要添加的机场货物的数量。

**样例**：# Connect to the AirSim simulator

client = airsim.MultirotorClient()

client.confirmConnection()

loc = airsim.Vector3r(1, 1, 0)  
 client.addLandingPort('port', loc, 7)

（7）VehicleClient类的方法delLandingPort(self, port\_name)。

功能：删除一个机场。

参数：port\_name：要删除的机场名称。

（8）VehicleClient类的方法getBoxesNumber(self, port\_name)。

功能：获取一个机场的货物数量。

参数：port\_name：要查询的机场名称。

输出：int，机场货物数量。

（9）VehicleClient类的方法setBoxesNumber(self, port\_name, box\_num)。

功能：设置一个机场的货物数量。

参数：port\_name：要设置的机场名称；

box\_num：要设置的货物数量。

（10）VehicleClient类的方法addFlightPip(self, name, pose, scale, color, cube = True)。

功能：添加一条航路管道。

参数：name：要添加的管道名称；

pose(Pose)：要添加的管道姿态；

scale(Vector3r)：要添加的管道尺寸；

color(list)：要添加的管道颜色；

cube(bool)：要添加的管道形状，True（四棱锥）、False（圆柱）。

**样例**：# Connect to the AirSim simulator

client = airsim.MultirotorClient()

client.confirmConnection()

cir\_scale = airsim.Vector3r(40, 40, 20)  
 cir\_color = [0.976, 0.973, 0.422, 0.65]

# 生成每个节点的名称，如 'circle1', 'circle2', 等  
 circle\_name = "circle1"  
 # 设置节点的位置  
 cir\_pose = airsim.Pose(airsim.Vector3r(0, 0, -120), airsim.Quaternionr(0, 0, 0, 1))

# 调用 addFlightPip 函数，生成圆柱形管道  
 client.addFlightPip(circle\_name, cir\_pose, cir\_scale, cir\_color, cube=False)

（11）VehicleClient类的方法delFlightPip(self, name)。

功能：删除一条航路管道。

参数：name：要删除的管道名称。

1. 交通工具的基本控制指令

（1）MultirotorClient类的方法takeoffAsync(self, timeout\_sec = 20, vehicle\_name = '')

功能：控制无人机起飞

参数：vehicle\_name：所要控制的无人机的名称。

（2）MultirotorClient类的方法landAsync(self, timeout\_sec = 60, vehicle\_name = '')

功能：控制无人机降落

参数：vehicle\_name：所要控制的无人机的名称。

（3）MultirotorClient类的方法moveByVelocityAsync(self, vx, vy, vz, duration, drivetrain = DrivetrainType.MaxDegreeOfFreedom, yaw\_mode = YawMode(), vehicle\_name = '')

功能：通过控制速度来控制无人机移动

主要参数：vx, vy, vz：三个坐标轴方向的速度分量；

duration：以该速度飞行的持续时间；

vehicle\_name：所要控制的无人机的名称。

（4）MultirotorClient类的方法moveToPositionAsync(self, x, y, z, velocity, timeout\_sec = 3e+38, drivetrain = DrivetrainType.MaxDegreeOfFreedom, yaw\_mode = YawMode(), lookahead = -1, adaptive\_lookahead = 1, vehicle\_name = '')

功能：控制无人机移动到目标点

主要参数：x, y, z：无人机移动的目标点的三个坐标轴方向的位置分量；

velocity：无人机飞行速度；

vehicle\_name：所要控制的无人机的名称。

（5）VehicleClient类的方法pickBox(self, drone\_name, port\_name)。

功能：控制无人机在指定机场取货。

参数：port\_name：取货的机场名称；

drone\_name：取货的无人机名称。

（6）VehicleClient类的方法dropBox(self, drone\_name, port\_name)。

功能：控制无人机在指定机场放货。

参数：port\_name：放货的机场名称；

drone\_name：放货的无人机名称。

（7）VehicleClient类的方法getPickedStatus(self, drone\_name)。

功能：查询无人机是否携带货物。

参数：drone\_name：查询的无人机名称。

输出：bool，True（携带货物）、False（未携带货物）。

（8）MultirotorClient类的方法getMultirotorState(self, vehicle\_name = '')。

功能：获取无人机的状态。

参数：drone\_name：查询的无人机名称。

输出：无人机的状态MultirotorState类，包含以下信息：

collision = CollisionInfo()  
 kinematics\_estimated = KinematicsState()  
 gps\_location = GeoPoint()  
 timestamp = np.uint64(0)  
 landed\_state = LandedState.Landed  
 rc\_data = RCData()  
 ready = False  
 ready\_message = ""  
 can\_arm = False

**样例**：import airsim

# Connect to the AirSim simulator

client = airsim.MultirotorClient()

client.confirmConnection()

# 定义位置

position = airsim.Vector3r(0, 0, 0)

# 定义方向

orientation = airsim.to\_quaternion(0, 0, 0, 1)

pose = airsim.Pose(position, orientation)

# 创建一个名字叫做drone、类型为simplefight、姿态为pose的无人机

client.simAddVehicle(vehicle\_name=“drone”,vehicle\_type= “simplefight”, pose=pose)

# 启用控制接口

client.enableApiControl(True, vehicle\_name='drone')  
 client.armDisarm(True, vehicle\_name='drone')

# 添加一个叫做”port”的机场

loc = airsim.Vector3r(0, 0, 0)  
 client.addLandingPort('port', loc, 7)

# 起飞

client.takeoff(vehicle\_name='drone')

# 飞行到机场上方

client.moveToPositionAsync(0, 0, -5, 10, vehicle\_name='drone').join()

# 取货

client.pickBox('drone', port).join()

# 以(1,1,-5)的速度飞行2s，再回到机场上方

client.moveByVelocityAsync(1, 1, -5, 2, vehicle\_name='drone').join()

client.moveToPositionAsync(0, 0, -5, 10, vehicle\_name='drone').join()

# 放货

client.dropBox(‘drone’, port).join()

# 降落

client.landAsync(vehicle\_name='drone')