1、实验名称及目的

Imu 和相机数据获取实验：通过 python 接口获取 Imu 和相机数据。

2、实验原理

首先进行 Config.json 文件进行相机传感器的配置，配置参数如下解释：

“SeqID”代表第几个传感器。此处 0、1、2 分别表示第 1、2、3 个传感器（免费版只支持 2 个图）。

“TypeID”代表传感器类型 ID，1:RGB 图（免费版只支持 RGB 图），2:深度图，3:灰度图。

“TargetCopter”传感器装载的目标飞机的 ID，可改变。

“TargetMountType”代表坐标类型，0：固定飞机上（相对几何中心），1：固定飞机上（相对底部中心），2：固定地面上（监控）也可变。

“DataWidth”为数据或图像宽度此处为 640，“DataHeight”为数据或图像高度此处为 480。

“DataCheckFreq”检查数据更新频率此处为 30HZ。

“SendProtocol[8]”为传输方式与地址， SendProtocol[0]取值 0：共享内存（免费版只支持共享内存），1：UDP 直传 png 压缩，2：UDP 直传图片不压缩，3：UDP 直传 jpg 压缩；SendProtocol[1-4]：IP 地址； SendProtocol[5]端口号。

“CameraFOV”为相机视场角（仅限视觉类传感器），单位度也可改变。

“SensorPosXYZ[3]”为传感器安装位置，单位米也可改变。

“SensorAngEular[3]”为传感器安装角度，单位度°也可改变。

深度相机输出的数据是以 uinit16 存储和传输的，它的数据范围是 0~65535。默认情况下，一个单位表示 1mm（由 otherParams[2]控制），也就是说最大范围是 0 到 65.535 米。但是，数据范围并不代表相机的实际探测距离，还需要 otherParams[0]设置最小探测距离 otherParams[1]设置最大探测距离。otherParams[0]：深度相机的最小识别距离（单位米），如果深度距离小于本值，那么输出 NaN 对应 65535。otherParams[1]：深度相机的最大识别距离（单位米），如果深度距离大于本值，那么输出 NaN 对应 65535。OtherParams[2]：深度相机 uint16 输出值的刻度单位（单位米），默认情况下深度值以毫秒为单位，因此需要填 0.001。注，默认值填 0 的话，会被替换为 otherParams[2]=0.001。实际深度值（单位米）＝深度图片值（uint16 范围）* otherParams[2]。

然后通过平台的接口 vis.sendReqToUE4() 向 RflySim3D 发送取图请求，并通过接口 vis.startImgCap()开启取图操作。并进行对图像的获取与操作。

UE4CtrlAPI 是 UE 相关的接口，其中包含的 sendUE4Cmd，其输入不同将会对 UE 产生不同的作用可选输入如下

1. RflyShowTextTime(String txt, float time)\\ 让 UE 显示 txt，持续 time 秒
2. RflyShowText(String txt)\\ 让 UE 显示 txt，持续 5 秒
3. RflyChangeMapbyID(int id)\\ 根据地图的 ID 切换 RflySim3D 场景地图

4. RflyChangeMapbyName(String txt)\\ 根据地图名切换 RflySim3D 场景地图

5. RflyChangeViewKeyCmd(String key, int num) \\ 与在 RflySim3D 中按一个 key + num 效果一致

6. RflyCameraPosAngAdd(float x, float y, float z,float roll,float pitch,float yaw) \\ 给摄像机的位置与角度增加一个偏移值

7. RflyCameraPosAng(float x, float y, float z, float roll, float pitch, float yaw) \\ 设置摄像机的位置与角度(UE 的世界坐标)

8. RflyCameraFovDegrees(float degrees) \\ 设置摄像机的视域体 FOV 角度

9. RflyChange3DModel(int CopterID, int veTypes=0) \\ 修改一个无人机的模型样式

10. RflyChangeVehicleSize(int CopterID, float size=0) \\修改一个无人机的缩放大小

11. RflyMoveVehiclePosAng(int CopterID, int isFitGround, float x, float y, float z, float roll, float pitch, float yaw) \\ 给无人机的位置与角度设置一个偏移值，isFitGround 设置无人机是否适应地面

12. RflySetVehiclePosAng(int CopterID, int isFitGround, float x, float y, float z, float roll, float pitch, float yaw) \\设置无人机的位置与角度

13. RflyScanTerrainH(float xLeftBottom(m), float yLeftBottom(m), float xRightTop(m), float yRightTop(m), float scanHeight(m), float scanInterval(m)) \\ 扫描地形，生成一个 png 的高度图与 txt，CopterSim 程序会需要它才知道 UE 有哪些地形、以及它们的高程

14. RflyCesiumOriPos(double lat, double lon, double Alt) \\ 根据经纬度修改 Cesium 的原点位置

15. RflyClearCapture()\\ 清空抓取的图像

16. RflySetActuatorPWMs(int CopterID, float pwm1, float pwm2, float pwm3, float pwm4, float pwm5, float pwm6, float pwm7, float pwm8); \\传入 8 个值，并触发目标无人机的蓝图的接口函数

17. RflySetActuatorPWMsExt(int CopterID, float pwm9, float pwm10, float pwm11, float pwm12, float pwm13, float pwm14, float pwm15, float pwm16, float pwm17, float pwm18, float pwm19, float pwm20, float pwm21, float pwm22, float pwm23, float pwm24);\\传入 16 个值，并触发目标无人机的蓝图接口函数。该函数需要完整版才能有作用

18. RflyReqVehicleData(FString isEnable);如果’isEnable’不为 0，则 RflySim3D 会开始发送所有 Copter 的数据（就是前面介绍的 reqVeCrashData）。

19. RflySetPosScale(float scale);\\全局位置的缩放

20. RflyLoad3DFile(FString FileName);\\加载并执行路径下的 TXT 脚本文件

21. RflyReqObjData(int opFlag, FString objName, FString colorStr);\\请求获取三维场景中物体的数据

22. RflySetIDLabel(int CopterID, FString Text, FString colorStr, float size);\\设置一个 Copter 的头顶 ID 显示内容（默认显示 CopterID）

23. RflySetMsgLabel(int CopterID, FString Text, FString colorStr, float size, float time, int flag);\\设置一个 Copter 头顶的 Message 显示的内容

24. RflyDelVehicles(FString CopterIDList);\\删除一些 Copter（逗号是分隔符）

25. RflyDisableVeMove(FString CopterIDList, int disable);\\拒接接收指定 ID 的 Copter 的信息（逗号是分隔符）

26. 除此之外，还有一些 UE 内置的命令可以使用，例如‘stat fps’可以显示当前帧率，‘t.Maxfps 60’可以设置最大帧率为 60。

3、实验效果

本实验通过 python 接口设置 Imu 和相机频率，并获取数据。

4、文件目录

文件夹/文件名称	说明
VisionCapAPIDemo.bat	启动仿真配置文件
VisionCapAPIDemo.py	Python 实验脚本
Config.json	视觉传感器配置文件
log	Imu 和相机数据文件夹

5、运行环境

序号	软件要求	硬件要求	
		名称	数量(个)
1	Windows 10 及以上版本	笔记本/台式电脑①	1
2	RflySim 平台免费版及以上		
3	Visual Studio Code		

① ：推荐配置请见：https://doc.rflysim.com/1.1InstallMethod.html

6、实验步骤

Step 1：

修改 VisionCaptureApi.py。在 img_mem_thrd 函数内，“self.timeStmp[idx]=”语句后，添加时间戳打印语句；在 getIMUDataLoop 函数内，“self.imu.timestmp=”语句后，添加时间戳打印语句。

Step 2：

修改 VisionCapAPIDemo.py 文件，“ue.sendUE4Cmd('t.MaxFPS90',0)”语句，其中的 90 可以替换成希望 UE4 运行的帧率。

注意：从 UE 取图都会有一个 Tick（例如 90Hz 就是 1/90.0 秒）的延迟，也就是在下一帧才能拿到图片并发送。因此，UE 的运行帧率决定了图像取图延迟的最小值（还需要加入数据传输等延迟）

Step 3：

修改 Config.json 文件，DataCheckFreq 的条目对应了图片的取图和发送频率。

1）在本例中，UE 的 MaxFPS 设置为 90Hz，而 DataCheckFreq 取图 30Hz，则 UE 会每 3 个 Tick 发送一次图片，最小延迟是 1/90 每秒。

2）如果设置 UE 的 MaxFPS 设置为 30hz，而取图 DataCheckFreq 取图 30Hz，则最小延迟是 1/30Hz。

3）如果设置 UE 的 MaxFPS 设置为 100hz，而取图 DataCheckFreq 取图 10Hz，则最小延迟是 1/100Hz。

4）如果设置 UE 的 MaxFPS 设置为 10hz，而取图 DataCheckFreq 取图 10Hz，则最小延迟是 1/10Hz

从上面分析可见

1）MaxFPS 尽量取更大值，能显著减小取图延迟，但是对显卡的考验会越来越大。

2）DataCheckFreq 的取图频率应该尽量是 MaxFPS 的公因数，可以整除，这样能够得到稳定的取图频率。

VisionCapAPIDemo.py 中“vis.sendImuReqCopterSim(1)”语句的意思是向 CopterSim 请求 IMU 的数据，采用的默认频率为 200Hz。

1)IMU 的时间戳和取图时间戳都是用的 CopterSim 开始仿真后的时间。IMU 的时间戳是从 CopterSim 直接通过 UDP 读取，几乎没有延迟。

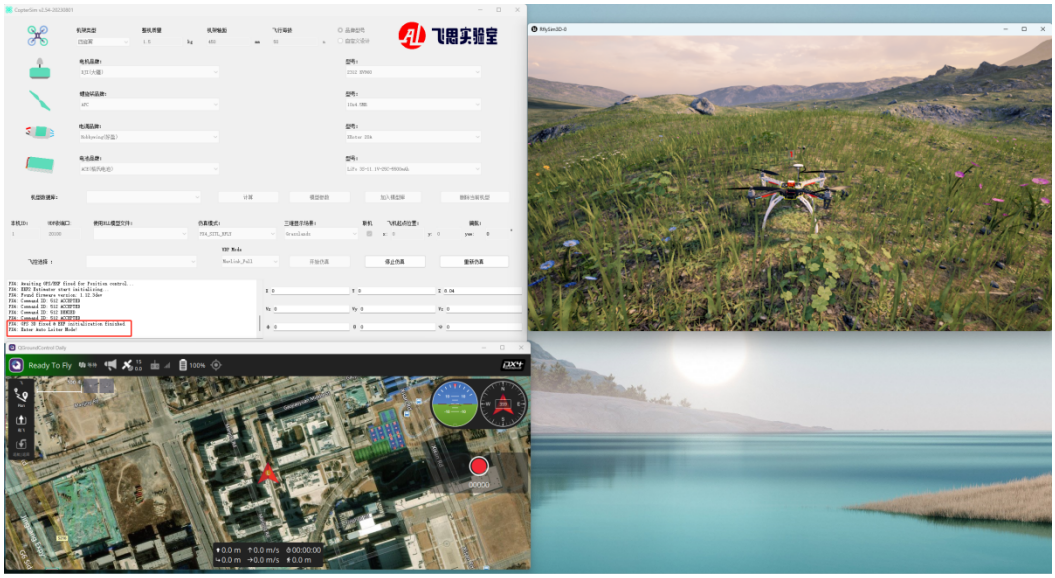
2）Python 的取图时间戳，是 CopterSim 先发送给 RflySim3D（约 50Hz 或 100Hz），RflySim3D 经过一个 Tick 的延迟，再随着图片一起转发到本 Python 程序。

3)通过图片时间戳与 IMU 时间戳的对比，可以大致判断取图环境的延迟情况。

log\UE90Hz-Cpature30Hz-IMU200hz.txt 对应了本 demo 的记录数据，通过 Capture：的图像时间戳与最近的 IMU 时间戳对比，可知本取图频率稳定在 30Hz，且取图延迟在 0.01~0.015s 左右。这个时间延迟已经非常小了（小于人眼的反应时间），因此是可以用于控制无人机的高机动飞行的。

Step 4：

以管理员方式运行 VisionCapAPIDemo.bat 开启一个飞机的软件在环仿真。将会启动 1 个 QGC 地面站，1 个 CopterSim 软件且其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，并且 RflySim3D 软件内有 1 架无人机。



Step 5：

用 VScode 打开到本实验路径文件夹， 运行 VisionCapAPIDemo.py 文件，待起飞之后，在 SITL 的黑窗口按下任意键，关闭 CopterSim 和 RflySim3D，此时 python 内数据会停止更新，将数据拷贝到一个 txt 文件里面。

Step 6：

在下图 VScode 中，点击“终止终端”，可以彻底退出脚本运行。

7、参考文献

[1]. 无

8、常见问题

Q1：无

A1：无