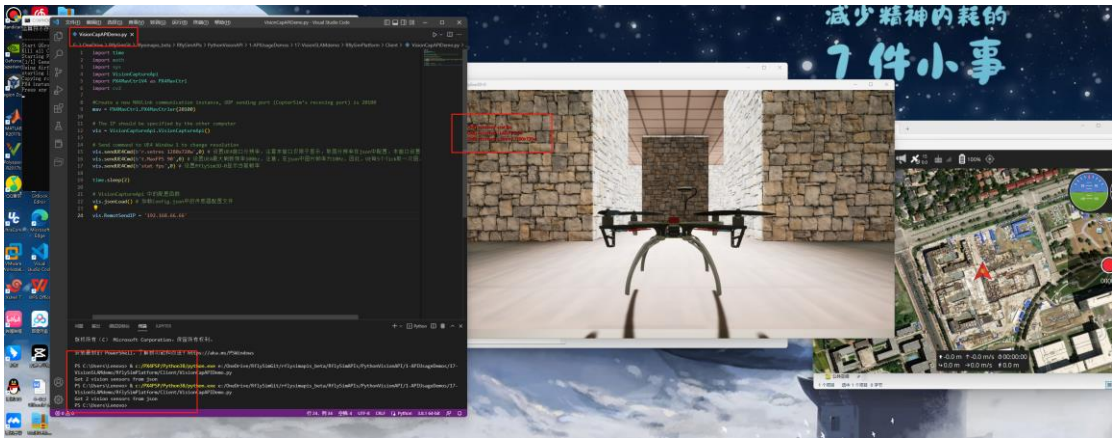
1.打开 RflySimAPIs\PythonVisionAPI\1-APIUsageDemos\17-VisionSLAMdemo\RflySimPlatform\Client 已管理员方式运行 VisionCapAPIDemo.bat，等待 3DFixed 之后，再用 Python 运行 VisionCapAPIDemo.py，待起飞之后，在 SITL 的黑窗口按下任意键，关闭 CopterSim 和 RflySim3D，此时 Python 内数据会停止更新，将数据拷贝到一个 txt 文件里面，进行后续的分析。运行效果如下图所示：



注意事项：

1.修改 VisionCaptureApi.py，打印图片和 IMU 的时间戳

1）img_mem_thrd 函数，“self.timeStmp[idx]=”语句后，添加时间戳打印语句

2）getIMUDataLoop 函数，“self.imu.timestmp=”语句后，添加时间戳打印语句

2.修改 VisionCapAPIDemo.py 文件，“vis.sendUE4Cmd(b'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）

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

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。

4.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。

5.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。6.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。