1.实验名称及目的

1.1实验名称

RflySim鱼眼相机实验

1.2 实验目的

本实验通过本例程展示如何在 RflySim 平台中使用鱼眼相机（fisheye）进行仿真视觉获取与飞控联动：

- 熟悉并修改 `Config.json` 中的视觉传感器配置项（包含 `TypeID`、`DataWidth`、`DataHeight`、`CameraFOV`、`otherParams` 等）。
- 使用 `VisionCaptureApi` 从仿真端获取鱼眼图像并在本地通过 OpenCV 可视化。
- 使用 `PX4MavCtrlV4` 发起 MAVLink 连接，切换到 offboard 模式并发送基本位置命令（例程演示 SendPosNED）。
- 分析典型鱼眼成像与配置对图像质量、畸变和分辨率的影响，并掌握常见问题的排查方法。

完成后应能独立运行 `SITLPosStr.bat` 启动仿真、运行 `example.py` 观测画面，并通过修改 `Config.json` 验证配置项变更的效果。

1.3 关键知识点（结合 `example.py` 的逐步解析）

下面将 `example.py` 中的主要逻辑拆解为关键知识点，便于快速理解与复用：

1. VisionCaptureApi（视觉采集链路）
   - 创建与配置：`vis = VisionCaptureApi.VisionCaptureApi()`，调用 `vis.jsonLoad()` 读取 `Config.json` 中的传感器配置（`VisionSensors`）。
   - 请求与启动：`vis.sendReqToUE4()` 向仿真端发送取图请求并做握手验证，`vis.startImgCap()` 启动图像采集/共享内存读取。若 `sendReqToUE4()` 返回 False，程序会退出（表明仿真端未就绪或配置不匹配）。

- 获取图像：主循环中通过 `vis.hasData[0]` 判断是否有新帧，通过 `vis.Img[0]` 读取图像数组，可直接交给 OpenCV 显示。注意索引对应配置中的传感器序号（例如 `SeqID`）。

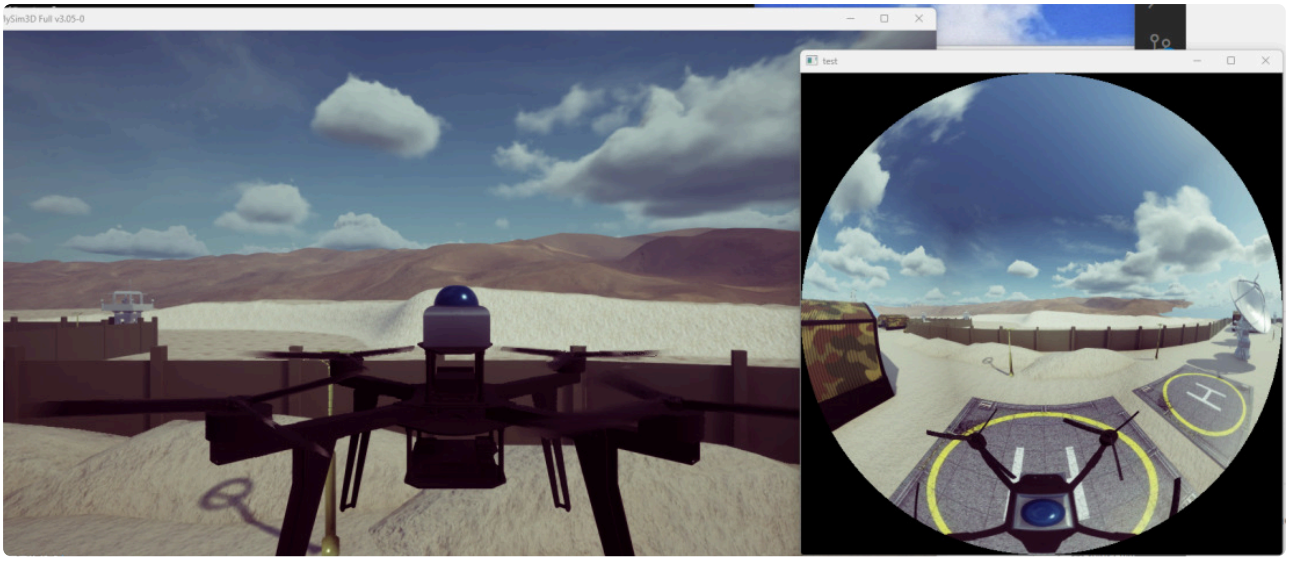
2. PX4MavCtrlV4（飞控控制链路）
   - 初始化：`mav = PX4MavCtrl.PX4Mctrler(1)`，`mav.InitMavLoop()` 启用 MAVLink 接收循环，通常需等待几秒以建立链路。
   - 模式与上锁：`mav.initOffboard()` 切换 offboard 模式，`mav.SendMavArm(True)` 解锁飞机。
   - 位置控制示例：`mav.SendPosNED(0,0,-10,0)` 将无人机移动到 NED 坐标系下的目标位置（此处示例为相对高度 -10m）。实际控制时可替换为速度或姿态接口。
3. OpenCV 可视化与主循环
   - 调用 `cv2.imshow("test", vis.Img[0])` 展示图像，`cv2.waitKey(1)` 处理窗口消息。
   - 采用 `time.sleep(0.03)` 控制循环频率（示例贴近 30Hz），同时避免忙等待。
4. 配置与鱼眼成像要点
   - 鱼眼模型：RflySim 的鱼眼近似采用等距投影（与常见的 Kannala–Brandt 多项式近似一致），畸变校正与重投影时需使用相应模型与内参。
   - 分辨率与拉伸：为避免图像被拉伸，`DataWidth` 和 `DataHeight` 应设置相等。
   - cubemap 与 otherParams：当鱼眼由 cubemap 构成时，`otherParams[0]` 用于指定单个贴图大小（0 表示使用默认值，例如 512）。

示例中涉及的代码文件名：`example.py`、`Config.json`、`SITLPosStr.bat`、以及模块 `VisionCaptureApi`、`PX4MavCtrlV4`、`UE4CtrlAPI`。

2.实验效果

运行 `SITLPosStr.bat` 启动仿真并执行 `example.py` 后，典型的实验效果包括：

- 本地弹出 OpenCV 窗口（示例标题为 "test"），实时显示来自鱼眼相机的图像帧（示例截图见 rfly_image3.png）。
- 图像为鱼眼视角（极宽视场），可以观察到强烈的径向畸变，这在 `Config.json` 中设置的 `CameraFOV` 与内参会直接影响畸变程度。
- 在程序启动时，例程会初始化 MAVLink 链路并尝试切换到 offboard 模式并解锁；在成功后会发送位置命令（示例为 `SendPosNED(0,0,-10,0)`），可以在仿真界面中看到无人机移动到目标高度。
- 图像刷新频率近似与 `Config.json` 中的 `DataCheckFreq`（或示例程序主循环节拍）一致，示例中采用约 30Hz 的循环节拍。



验证要点：

1. 确认已运行 `SITLPosStr.bat` 并且仿真端显示就绪界面（参见 rfly_image4.png）。
2. 运行 `example.py` 后，控制台不应提前退出；若 `sendReqToUE4()` 返回 False，说明视觉请求未成功建立（请检查 `Config.json` 与仿真端配置是否一致）。
3. 若图像出现拉伸或黑屏，检查 `DataWidth` 与 `DataHeight` 是否相等，以及 `TypeID` 是否正确设置为鱼眼类型（示例中为 8）。

3.文件目录

例程目录：[安装目录]\RflySimAPIs\8.RflySimVision\0.ApiExps\11.FishEyeDemo

文件夹/文件名称	说明
SITLPosStr.bat	软件在环启动脚本
example.py	py控制程序

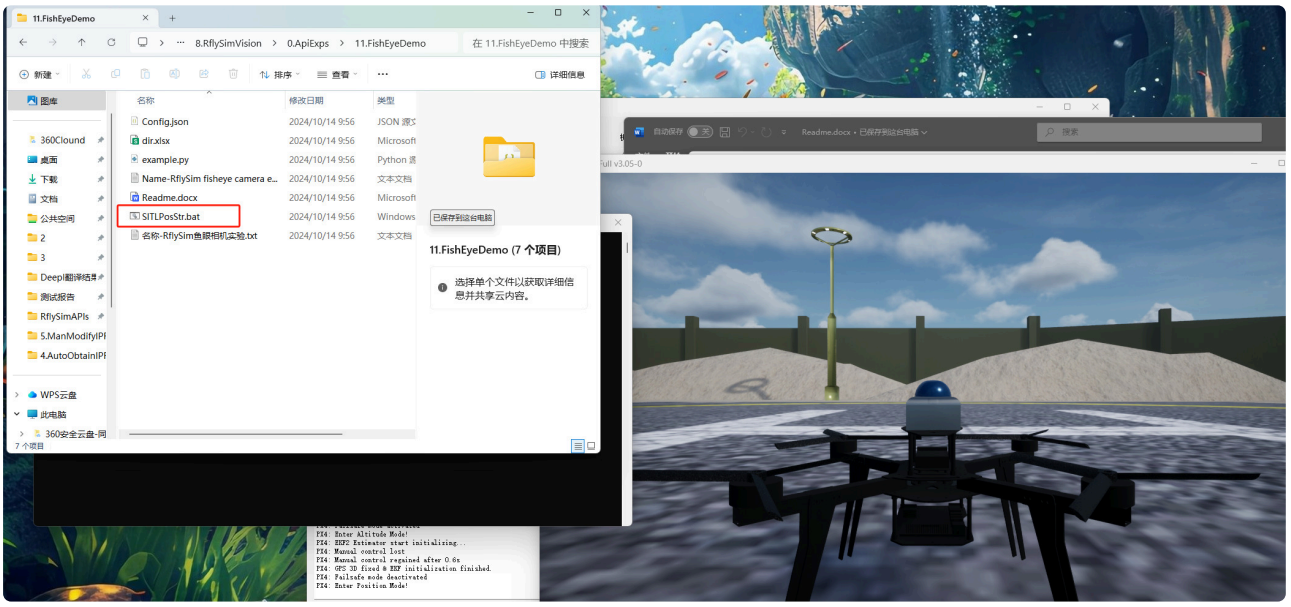
4.运行环境

序号	软件要求	硬件要求	
		名称	数量
1	Windows 10及以上版本	笔记本/ 台式电脑	1

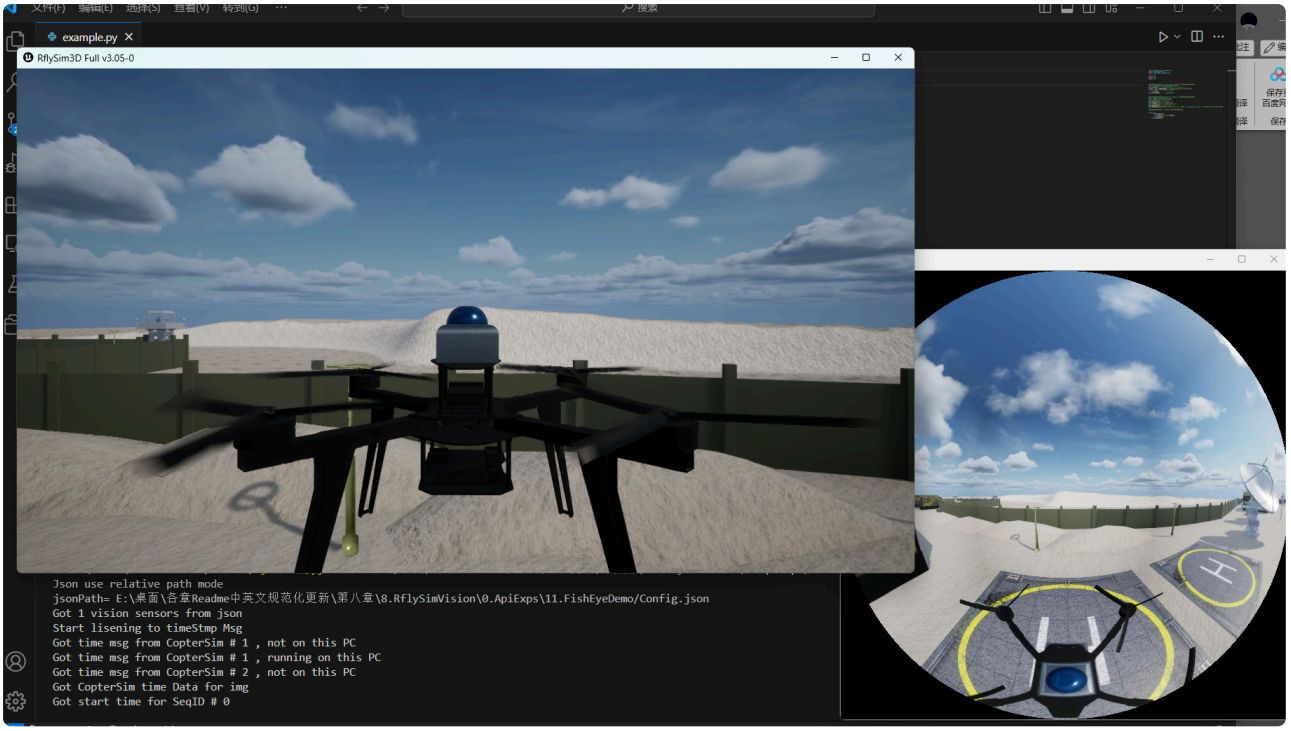
序号	软件要求	硬件要求	
2	RflySim工具链		
3	Visual Studio Code		
①：推荐配置请见：https://rflysim.com/doc/zh/1/InstallLearn.html			

5.实验步骤

1.运行SITLPosStr.bat 脚本启动平台，如下图所示：



2.双击启动Python38Run.bat在终端输入`python example.py`运行example.py 即可看到如下图所示的效果



6.参考资料

- Kannala, J. and Brandt, S. "A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-eye Lenses."（Kannala–Brandt 鱼眼模型论文）
- OpenCV 官方文档：Fish-eye 模块与相机标定（参考鱼眼畸变和重投影方法） — https://docs.opencv.org/
- RflySim 官方文档：安装与使用（包含传感器配置说明） — https://rflysim.com/doc/zh/

（以上资料用于理解鱼眼成像模型、标定方法以及 RflySim 中传感器配置的语义）

7.常见问题

Q1：运行 `example.py` 没有弹出图像窗口或窗口黑屏，控制台退出或程序结束。

A1：请按下列顺序检查：

1. 确认已先运行 `SITLPosStr.bat` 并且仿真端已启动；仿真端未就绪会导致 `sendReqToUE4()` 返回 False，从而程序退出。
2. 检查 `Config.json` 中的传感器条目是否存在且正确（字段如 `VisionSensors`、`SeqID`、`TypeID`、`DataWidth`、`DataHeight` 等）。

3. 在 `example.py` 中，观察 `vis.sendReqToUE4()` 的返回值；若为 False，说明握手失败，可在仿真端或日志中查看端口/协议匹配问题。

Q2：画面被拉伸或比例不正确。

A2：应确保 `DataWidth` 与 `DataHeight` 设置为相同值（示例中通常为 640×640）；若宽高不一致会造成拉伸。另请确认显示窗口的缩放不被外部工具强制改变。

Q3：图像分辨率与帧率不够理想如何调整?

A3：调整 `Config.json` 中的 `DataWidth`、`DataHeight` 以及 `DataCheckFreq`（或在 `example.py` 中调整主循环的 sleep 时间），注意提高分辨率会消耗更多带宽与计算资源。

Q4：如何改变鱼眼视场或畸变行为?

A4：通过修改 `Config.json` 中的 `CameraFOV` 和对应的相机内参来调整；若需后处理畸变校正，请参考 OpenCV 的 fish-eye 标定/去畸变接口并使用对应模型（Kannala–Brandt 与 OpenCV 支持的 fisheye 模型互相参照）。

Q5：无人机没有响应位置信令或无法切换到 offboard 模式。

A5：请确认 MAVLink 链路已建立（查看 `example.py` 中 `mav.InitMavLoop()` 是否成功），并检查飞控日志与仿真端是否接收到命令；防火墙或端口冲突也可能导致消息丢失。

Q6：我需要扩展示例以保存图像或按帧处理，如何改造?

A6：在主循环中当 `vis.hasData[0]` 为 True 时，可直接对 `vis.Img[0]` 进行 cv2.imwrite 保存或交给自定义处理函数；若需高并发写磁盘，建议使用独立线程/队列以避免阻塞主循环。

如果上述排查仍无效，请把控制台日志与相关文件（如 `Config.json` 与 `example.py` 的修改片段）整理后进一步分析。