仿真吊舱UI界面操作实验

1. 实验目的

本实验旨在通过RflySim仿真平台，实现一个完整的仿真吊舱UI控制系统，让用户能够：

1. 掌握仿真吊舱的基本操作原理和控制方法
2. 学习通过UI界面控制吊舱的俯仰角、偏航角、变焦、变倍等参数
3. 理解视觉传感器与仿真环境的交互机制
4. 掌握AI目标识别与追踪功能的实现原理
5. 熟悉RflySim API在无人机视觉仿真中的应用

2. 实验要求

- 软件要求：Windows 10及以上版本；RflySim工具链[1]。
- 硬件要求：笔记本/台式电脑1台[2]。

3.实验地址

例程目录：[安装目录]\RflySimAPIs\8.RflySimVision\0.ApiExps\12.UAV_PodSimUI

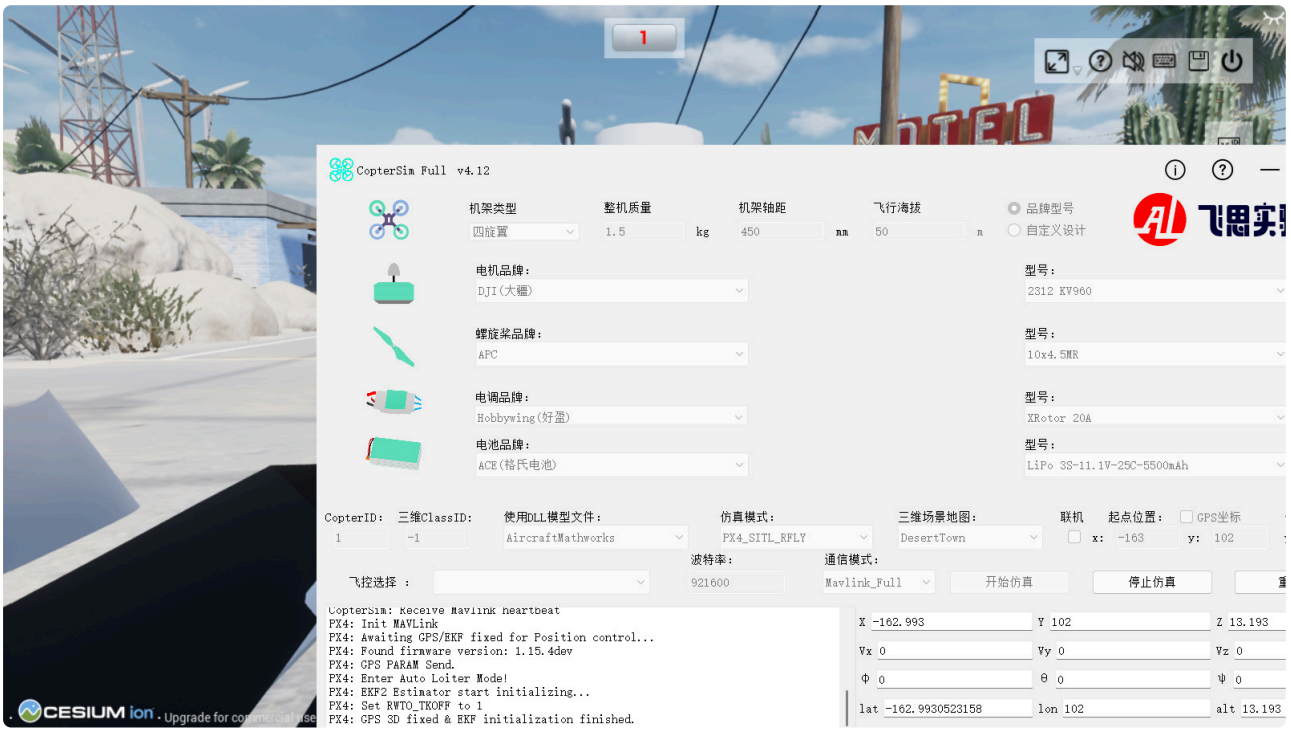
文件目录结构说明(使用代码块格式)：

```
├── AircraftMathworksSITLRun.bat        # 仿真环境启动脚本
├── CameraCtrlApi.py                    # 相机控制API模块
├── Config.json                         # 视觉传感器配置文件
├── controller.py                       # 主控制器模块
├── main.py                             # 程序主入口文件
├── Python38Run.bat                     # Python运行脚本
├── ui_mainwindow.py                    # UI界面模块
```

4. 实验内容或步骤

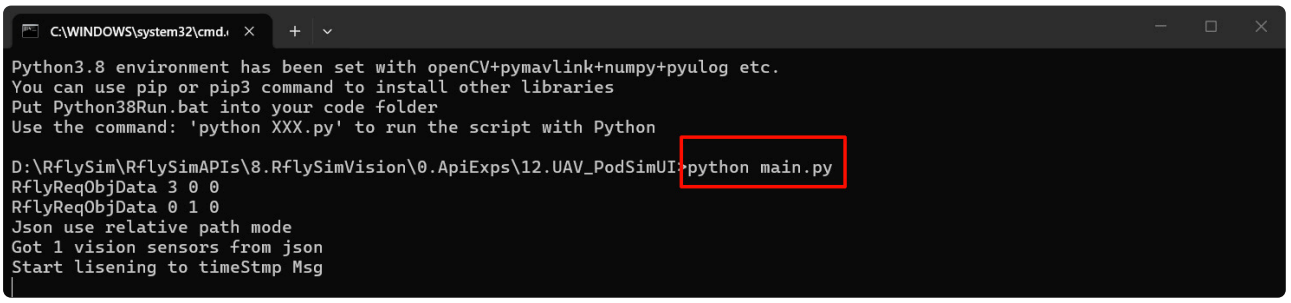
4.1 初始化仿真环境

双击运行`AircraftMathworksSITLRun.bat`文件，等待仿真环境初始化完成。脚本将会启动 1 个 CopterSim 、1 个 RflySim3D 软件。



4.2 启动仿真吊舱UI界面

双击 Python38Run.bat 文件，在弹出的对话框中输入：`python main.py`。



4.3 配置参数修改

根据实验需求，可修改`Config.json`文件中的视觉传感器参数：

- `DataWidth`和`DataHeight`：设置图像分辨率

- `CameraFOV`：调整摄像头视场角
- `SensorPosXYZ`：修改传感器安装位置坐标
- `SensorAngEular`：设置传感器初始姿态（弧度制）
- `SendProtocol`：配置数据传输协议和端口

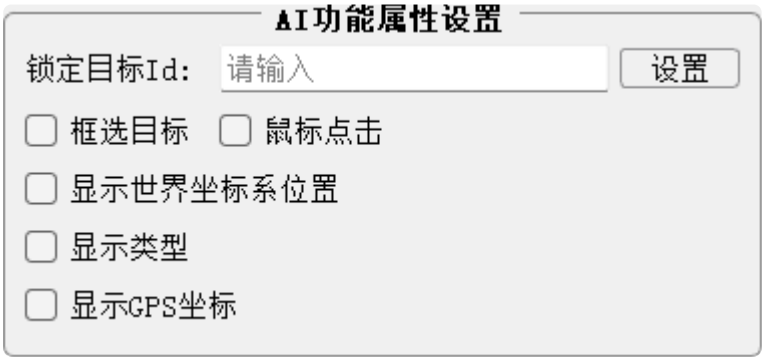
> **注意**：修改配置后需重启程序生效

4.4 UI界面操作

启动程序后，可通过以下方式控制仿真吊舱：

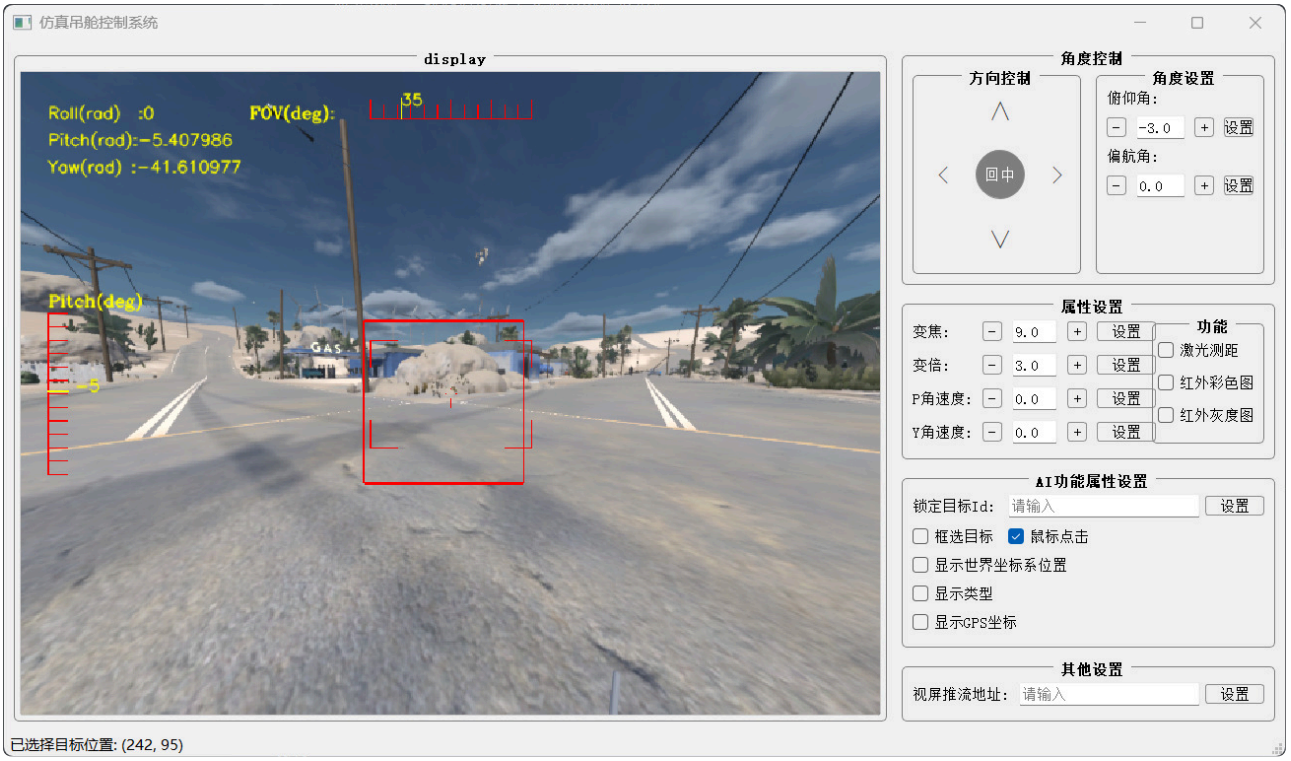
1. **角度控制**：
   - 使用方向键 `↑↓←→` 控制俯仰和偏航
   - 按 `Ctrl` 键配合方向键进行微调（步长0.1°）
   - 通过界面上的旋钮控件调节角度
2. **变倍控制**：
   - 按 `+` / `-` 键调整相机变倍值
3. **状态监控**：
   - 主界面实时显示当前角度、变倍值和连接状态
   - 日志区域记录操作指令和系统响应

4.5 AI功能属性设置



在右侧功能面板中（如图所示）：

- **锁定目标Id输入框**位于顶部，输入数字2-5可锁定对应ID的无人机
- **设置按钮**需在修改ID后点击确认（首次修改需先更改数字再点击）
- 界面实时显示当前锁定目标状态和吊舱视角



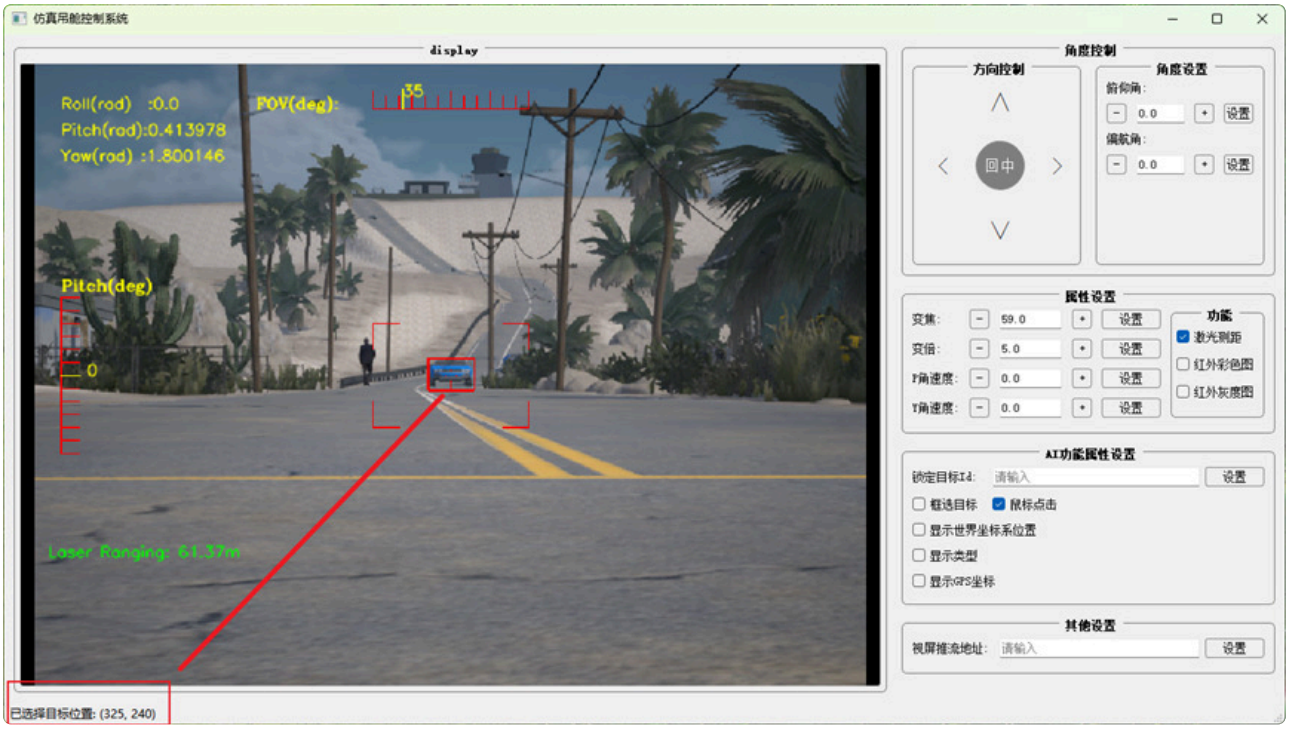
框选目标功能:

- 勾选复选框后，若已设置目标ID，系统会框选所有程序创建的无人机（ID 2-5）
- 若未设置ID，则框选指定ID的无人机
- 每次修改后需点击"设置"按钮生效（如图中红框所示）

鼠标点击功能:

- 勾选后可在图像窗口直接点击目标无人机（如图中鼠标指针位置）
- 点击成功时目标会被高亮框选（如图中黄色框）

- 若点击无响应：
  i. 减小视场角（调整Config.json中CameraFOV参数）
  ii. 放大画面重试
- 退出时需取消勾选（方向键控制将立即恢复）



> **操作提示**：当鼠标点击功能激活时（复选框勾选状态），方向键控制将失效。重新启用功能前需先取消勾选再重新勾选。

4.6 显示设置

- **显示世界坐标系**：勾选后，窗口图像上会显示锁定目标Id输入框内id为此数字的Copter的位置。
- **显示类型**：勾选后会会在图像上显示锁定目标Id输入框内id为此数字的Copter的Id。
- **显示GPS坐标**（开发中）。

5. 关键知识点

5.1 实验整体思路和框架

本实验采用MVC（Model-View-Controller）架构设计，实现了一个完整的仿真吊舱控制系统：

1. **模型层（Model）**：`CameraCtrlApi.py`和`KeyCtrl`类负责底层相机控制逻辑
2. **视图层（View）**：`ui_mainwindow.py`使用PyQt5构建用户界面
3. **控制层（Controller）**：`controller.py`中的`MainController`类协调模型和视图的交互
4. **主程序**：`main.py`作为程序入口，初始化整个应用

系统通过RflySim API与UE4仿真环境通信，实现实时图像传输和吊舱控制。

5.2 核心Python程序解析

5.2.1 main.py - 程序主入口

```python
import sys
from PyQt5 import QtWidgets
from ui_mainwindow import Ui_MainWindow
from controller import MainController

class MainWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super().__init__()

        # 设置UI
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        # 初始化控制器
        self.controller = MainController(self.ui, self)

        # 设置窗口标题
        self.setWindowTitle("仿真吊舱控制系统")

        # 初始化UI状态
        self.controller.update_status("应用程序已启动", "info")

    def closeEvent(self, event):
        """窗口关闭时清理资源"""
        self.controller.close()
        event.accept()

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)

    # 创建主窗口
    window = MainWindow()
    window.show()

    # 运行应用程序
    sys.exit(app.exec_())
```

功能说明:

- 创建PyQt5应用程序实例
- 初始化主窗口和控制器
- 设置窗口关闭事件处理
- 启动应用程序事件循环

5.2.2 controller.py - 主控制器模块

5.2.2.1 控制器初始化

```python
class MainController:
    def __init__(self, ui, main_window):
        self.ui = ui
        self.main_window = main_window

        # 初始化API
        self.ue = UE4CtrlAPI.UE4CtrlAPI()
        self.ue.initUE4MsgRec()
        self.vis = VisionCaptureApi.VisionCaptureApi()
        self.key_ctrl = CameraCtrlApi.KeyCtrl()
        self.img_ctrl = CameraCtrlApi.ImageCtrl()

        # 初始化相机状态
        self.angle_step = 1.0  # 默认角度步长
        self.init_velocity = 0.0  # 初始角速度
        self.init_magnification = 3.0  # 初始变倍值

        # 连接信号
        self.connect_signals()

        # 初始化UE4连接
        self.init_ue4()

        # 启动更新定时器
        self.timer = QtCore.QTimer()
        self.timer.timeout.connect(self.update_video)
        self.timer.start(33)  # ~30fps
```

5.2.2.2 UE4环境初始化

```python
def init_ue4(self):
    """初始化UE4连接"""
    self.ue.sendUE4PosScale(copterID=1, vehicleType=100, PosE=[-162, 102, -100])
    self.ue.sendUE4PosScale(
        copterID=2,
        vehicleType=100,
        PosE=[315, 250, -40],
        AngEuler=[0, 0, 1.57],
        Scale=[2, 2, 2],
    )  # 创建一架飞机
    # 创建3辆车
    self.ue.sendUE4PosScale(
        copterID=3, vehicleType=30, PosE=[-108, 100.4, 13], AngEuler=[0, 0, 0]
    )
    # ... 其他初始化代码

    # 加载传感器配置
    self.vis.jsonLoad()

    # 发送取图请求
    if not self.vis.sendReqToUE4():
        self.update_status("无法连接到UE4", "error")
        return False

    # 启动图像捕获
    self.vis.startImgCap()
    self.update_status("成功连接到UE4", "success")
    return True
```

5.2.2.3 视频更新循环

```python
def update_video(self):
    """更新UE4显示"""
    current_time = time.time()
    if current_time - self.last_update_time < 1/30.0:
        return

    self.last_update_time = current_time

    # 检查是否有新图像数据
    for y in range(len(self.vis.hasData)):
        if self.vis.hasData[y]:
            # 获取原始图像
            img = self.vis.Img[y]
            if img is None:
                continue

            # 绘制姿态信息
            roll, pitch, yaw = ang_eular
            self.img_ctrl.DisplayImg(img)
            self.img_ctrl.setRect(
                self.vis.VisSensor[y].DataWidth,
                self.vis.VisSensor[y].DataHeight
            )
            self.img_ctrl.drawCross()
            self.img_ctrl.drawCrossRect()

            # 绘制FOV和姿态信息
            self.img_ctrl.drawCoordinate(
                'FOV(deg):', 'top', fov, 15, 3, 0, 180, 10)

            # 3. 最终显示用RGB格式
            img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            height, width, _ = img.shape
            q_img = QtGui.QImage(
                img_rgb.data,
                width,
                height,
                width * 3,
                QtGui.QImage.Format_RGB888
            )
            self.ui.displayLabel.setPixmap(
                QtGui.QPixmap.fromImage(q_img).scaled(
                    self.ui.displayLabel.width(),
                    self.ui.displayLabel.height(),
                    QtCore.Qt.KeepAspectRatio
                )
            )
```

5.2.2.4 角度控制逻辑

```python
def adjust_angle(self, angle_type, direction):
    """调整角度"""
    try:
        # 切换到角度控制模式
        self.key_ctrl.AngOrAngV = 0
        self.key_ctrl.AngVel = [0.0, 0.0, 0.0]  # 重置角速度
        # 计算目标角度
        if angle_type == 'pitch':
            current = self.key_ctrl.AngEular[1]
            target = current + direction
            self.key_ctrl.AngEular[1] = target

        elif angle_type == 'yaw':
            current = self.key_ctrl.AngEular[2]
            target = current + direction
            self.key_ctrl.AngEular[2] = target

        # 更新UI显示
        self.update_ui_display()
        self.update_status(f"{angle_type.capitalize()} adjusting...", "info")

    except ValueError:
        self.update_status("Invalid value", "error")
```

5.2.3 CameraCtrlApi.py - 相机控制API

5.2.3.1 KeyCtrl类 - 键盘控制核心

```
class KeyCtrl:
    '''
        该类是通过键盘控制吊舱相机运动
            上(↑)下(↓)键控制俯仰角(pitch)；
            左(←)右(→)键控制偏航角(yaw);
            右Ctrl建 + 左(←)右(→) 控制横滚角(roll)
            焦距调节 alt+上, alt+下
    '''

    def __init__(self):
        '''
            增加参数
                mask_***: &比较, 0: 未激活  1: 激活
                press_**: 按键控制
                FOVOrFocalLength: 视场角/焦距        模式切换的时候会同步焦距/视场角
                CameraFocalLength: 焦距
                CmaeraFocalLengthBaseNum: 焦距步长
                AngOrAngV: 角度/角速度       切换至角度模式控制的时候会同步下当前角度
                AngVel: 角速度
                CurOZ: 当前变倍, 默认1
                OZBaseNum: 变倍基数
                TrackID: 追踪目标的ID
        '''

        self.mask_FOV = 1 << 1          # 视场角
        self.mask_Ang = 1 << 2          # 角度
        self.mask_Quat = 1 << 3         # 四元数
        self.mask_FocalLength = 1 << 4  # 焦距
        self.mask_BackCenter = 1 << 5  # 回中
        self.mask_AngVel = 1 << 6       # 角速度
        self.mask_OpticalZoom = 1 << 7 # 光学变倍
        self.mask_Track = 1 << 8        # 目标
        self.mask_Distance = 1 << 9     # 测距
        self.mask_SensorType = 1 << 10  # 传感器类型
        self.mask_AI = 1 << 11          # 自动框选目标

        self.AngOrAngV = 0              # 默认为角度控制
        self.AngEular = [0, 0, 0]       # 当前角度
        self.AngVel = [0, 0, 0]         # 角速度
        self.offset = 1                 # 每次偏移量1个degrees
```

5.2.3.2 ImageCtrl类 - 图像显示控制

```python
class ImageCtrl:
    def __init__(self, screenWidth=640, screenHeight=480) -> None:
        self.image = None
        self.centX = int(screenWidth/2)
        self.centY = int(screenHeight/2)
        #   十字星信息
        self.starMargin = 4  # 中心点间隔
        self.starWidth = 6  # 十字星的宽度
        self.lineWidth = 1  # 线的宽度
        self.lineColor = (0, 0, 255)
        self.valueColor = (0, 255, 255)

    def drawCross(self):
        cv2.line(self.image, (self.centX, self.centY-self.starMargin),
                 (self.centX, self.centY-self.starMargin), self.lineColor,
self.lineWidth)
        cv2.line(self.image, (self.centX, self.centY+self.starMargin),
                 (self.centX, self.centY+self.starMargin+self.starWidth),
self.lineColor, self.lineWidth)
        # ... 其他绘制代码
```

5.2.4 ui_mainwindow.py - 用户界面

5.2.4.1 UI布局设计

```python
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1300, 700)

        # 中央部件和主布局
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.mainLayout = QtWidgets.QHBoxLayout(self.centralwidget)

        # ========== 左侧视频区域 ==========
        self.displayGroup = QtWidgets.QGroupBox("display")
        self.displayGroup.setMinimumSize(800, 600)
        self.displayLayout = QtWidgets.QVBoxLayout(self.displayGroup)

        # 视频显示标签
        self.displayLabel = ClickableLabel()
        self.displayLabel.setAlignment(QtCore.Qt.AlignCenter)
        self.displayLabel.setStyleSheet("""
            background-color: #000000;
            color: #ffffff;
            border: 1px solid #cccccc;
            border-radius: 3px;
        """)
        self.displayLabel.setText("正在初始化...")
        self.displayLayout.addWidget(self.displayLabel)

        # ========== 右侧控制面板 ==========
        self.controlPanel = QtWidgets.QFrame()
        self.controlPanel.setMinimumWidth(350)
        self.controlLayout = QtWidgets.QVBoxLayout(self.controlPanel)

        # 角度控制组、属性设置组、AI功能组等...
```

5.3 关键技术点总结

1. **多线程图像处理**：使用QTimer定时器实现30fps的图像更新，避免UI阻塞
2. **位掩码控制**：使用位掩码技术实现多种控制模式的快速切换和状态管理
3. **坐标转换**：实现屏幕坐标到图像坐标的精确转换，支持鼠标点击目标选择
4. **RflySim API集成**：深度集成UE4CtrlAPI、VisionCaptureApi等RflySim核心API
5. **模块化设计**：将相机控制、图像显示、UI交互等功能模块化，便于维护和扩展
6. **实时状态同步**：实现UI状态与仿真环境状态的实时同步和反馈

6.参考资料

1. **RflySim官方文档**：https://rflysim.com/doc/zh/
2. **PyQt5官方文档**：https://www.riverbankcomputing.com/static/Docs/PyQt5/
3. **OpenCV官方文档**：https://docs.opencv.org/
4. **Python官方文档**：https://docs.python.org/zh-cn/3/
5. **UE4官方文档**：https://docs.unrealengine.com/

7.常见问题

Q1：程序启动后显示"无法连接到UE4"错误

现象：双击运行`Python38Run.bat`并输入`python main.py`后，程序启动但状态栏显示"无法连接到UE4"错误信息，视频区域显示"正在初始化..."但无图像显示。

原因分析：

1. RflySim仿真环境未正确启动
2. UE4与Python程序之间的网络连接问题
3. 配置文件`Config.json`中的端口设置错误
4. 防火墙或安全软件阻止了程序通信

解决方案：

1. **检查仿真环境**：确保已双击运行`AircraftMathworksSITLRun.bat`并等待CopterSim和RflySim3D完全启动
2. **验证网络连接**：检查Windows防火墙设置，确保允许Python和UE4程序通过防火墙
3. **检查配置文件**：确认`Config.json`中的`SendProtocol`端口设置正确，默认使用9999端口
4. **重启程序**：先关闭所有相关程序，然后按顺序重新启动：
   - 先运行`AircraftMathworksSITLRun.bat`
   - 等待仿真环境完全启动（约30-60秒）
   - 再运行`Python38Run.bat`并输入`python main.py`

Q2：方向键控制吊舱角度无响应

现象：程序正常启动并显示图像，但按方向键（↑↓←→）时吊舱角度不变化，UI界面上的角度数值也不更新。

原因分析：

1. 键盘焦点不在程序窗口上
2. AI功能中的鼠标点击模式被激活
3. 控制器信号连接失败
4. 键盘事件被其他程序拦截

解决方案：

1. **检查窗口焦点**：点击程序窗口确保其获得焦点（标题栏高亮显示）
2. **检查AI模式**：查看右侧控制面板中的"鼠标点击"复选框是否被勾选，如果勾选请取消
3. **验证信号连接**：在 `controller.py` 的 `connect_signals()` 方法中检查方向键按钮的信号连接
4. **关闭冲突软件**：关闭可能拦截键盘输入的其他软件（如游戏手柄驱动、键盘宏软件等）
5. **使用UI按钮**：尝试使用界面上的方向控制按钮（∧∨<>）进行测试

Q3：鼠标点击目标选择功能无效

现象：勾选"鼠标点击"复选框后，在图像窗口点击无人机目标，但目标没有被框选，吊舱也没有追踪点击的目标。

原因分析：

1. 视场角过大导致目标识别困难
2. 坐标转换计算错误
3. AI框选模式设置不正确
4. 目标ID未正确设置

解决方案：

1. **调整视场角**：修改 `Config.json` 中的 `CameraFOV` 参数，减小视场角（如从90°改为60°）
2. **放大画面**：使用变焦功能放大图像，使目标更清晰
3. **检查坐标转换**：确保 `handle_mouse_click` 方法中的坐标转换逻辑正确
4. **设置目标ID**：在锁定目标Id输入框中输入有效的目标ID（2-5）

5. **验证AI标志**：检查`AIFlag_Mouse`变量是否正确设置为1

Q4：图像显示卡顿或延迟严重

现象：视频显示区域图像更新缓慢，有明显的卡顿现象，操作响应延迟。

原因分析：

1. 计算机性能不足
2. 图像分辨率设置过高
3. 网络传输延迟
4. 程序更新频率设置不合理

解决方案：

1. **降低图像分辨率**：修改`Config.json`中的`DataWidth`和`DataHeight`参数，如从1280×720降低到640×480
2. **调整更新频率**：在`controller.py`中调整`self.timer.start(33)`的参数，增加更新间隔（如改为50ms）
3. **优化计算机性能**：关闭不必要的后台程序，确保有足够的内存和CPU资源
4. **检查网络设置**：如果使用UDP传输，检查网络带宽和延迟

---

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