基于ESDF和Voronoi图的无人机路径规划实验

1. 实验目的

通过处理二维栅格地图，构建包含障碍物距离信息的ESDF场和拓扑骨架的Voronoi图。基于这两者进行路径规划，并将规划轨迹映射到RflySim平台中，控制无人机实现闭环避障运动。

2. 实验要求

此编写实验目的

- 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\2.AdvExps\e16_ESDFPathPlan

文件目录结构说明(使用代码块格式)：

```
├── Readme.md                    # 项目根目录说明文档
├── WinWSL.bat                   # Windows WSL环境启动脚本
├── WslGUI.bat                   # WSL图形界面启动脚本
├── Client/                      # 客户端文件夹
│   ├── client_ue4.py            # UE4客户端Python脚本
│   ├── Config.json              # 配置文件
│   ├── Python38Run.bat          # Python 3.8运行脚本
│   ├── Readme.pdf               # 客户端说明文档
│   ├── readme.txt               # 客户端说明文本
│   ├── VisionCapAPIDemo.bat     # 视觉捕获API演示启动脚本
│   └── VisionCapAPIDemo.py      # 视觉捕获API演示Python脚本
├── maps/                        # 地图文件夹
│   ├── auto_crop_map.py         # 地图自动裁剪脚本
│   ├── map_cropped.pgm          # 裁剪后的地图图像
│   ├── map_cropped.yaml         # 裁剪后的地图配置文件
│   ├── map.pgm                  # 原始地图图像
│   ├── map.yaml                 # 原始地图配置文件
│   ├── OGM2ESDF.py              # ESDF地图转换和路径规划Python脚本
│   ├── planned_path.npy         # 规划路径数据文件
│   ├── WinWSL.bat               # WSL环境启动脚本
│   └── old_data/                # 旧数据文件夹
│       ├── map_cropped.pgm      # 旧裁剪地图
│       ├── map_cropped.yaml     # 旧裁剪地图配置
│       ├── my_map.pgm           # 自定义地图
│       └── my_map.yaml          # 自定义地图配置
└── Server/                      # 服务器文件夹
    ├── Config.json              # 服务器配置文件
    ├── esdf_planner.py          # ESDF路径规划器Python脚本
    ├── planned_path.npy         # 规划路径数据文件（复制使用）
    ├── tf_cfg.yaml              # TF坐标变换配置文件
    └── WinWSL.bat               # WSL环境启动脚本
```

4. 实验内容或步骤

> 本实验开始前，请先进行学习`[安装目录]\8.RflySimVision\2.AdvExps\e6_LaserSLAMdemo\Readme.pdf`实验步骤并完成单线激光雷达建图实验。

4.1 飞行点绘制

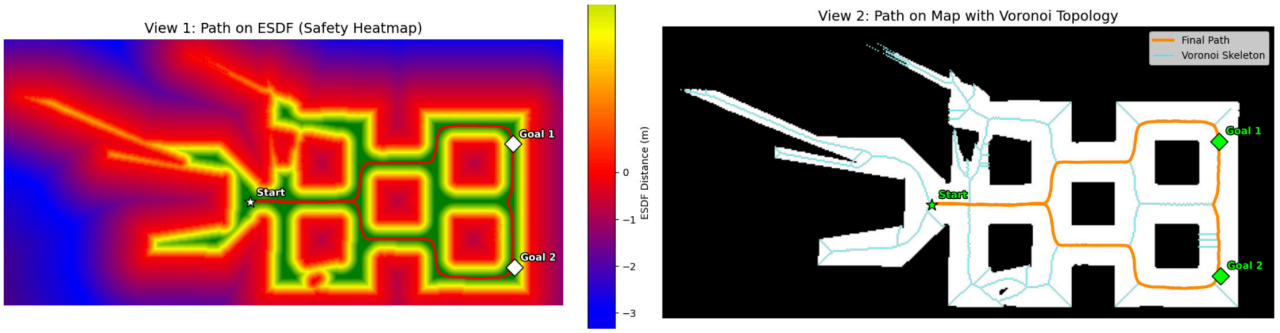
双击`maps\WinWSL.bat`脚本在弹出的CMD对话框中，输入`python3 OGM2ESDF.py`按下回车键，即可开始运行OGM2ESDF.py程序。

```
root@Byzeal: /mnt/e/PX4PSP/
root@Byzeal:/mnt/e/PX4PSP/RflySimAPIs/8.RflySimVision/2.AdvExps/基于ESDF图的路径规划/maps# python3 OGM2ESDF.py
/root/.local/lib/python3.10/site-packages/matplotlib/projections/__init__.py:63: UserWarning: Unable to import Axes3D. T
his may be due to multiple versions of Matplotlib being installed (e.g. as a system package and as a pip package). As a
result, the 3D projection is not available.
  warnings.warn("Unable to import Axes3D. This may be due to multiple versions of "
>>> 阶段 1: 地图处理
 - 计算 ESDF...
 - 提取 Voronoi...

>>> 阶段 2: 交互式选点 (闭环规划)
请在弹出的地图窗口中依次点击 3 个点:
  1. 起点 (Start)
  2. 目标点 1 (Goal 1)
  3. 目标点 2 (Goal 2)
程序将规划: 起点 -> 目标1 -> 目标2 -> 起点
```

根据上述窗口中的提示，在弹出的窗口中点击三个点依次为起点、目标的1、目标点2。

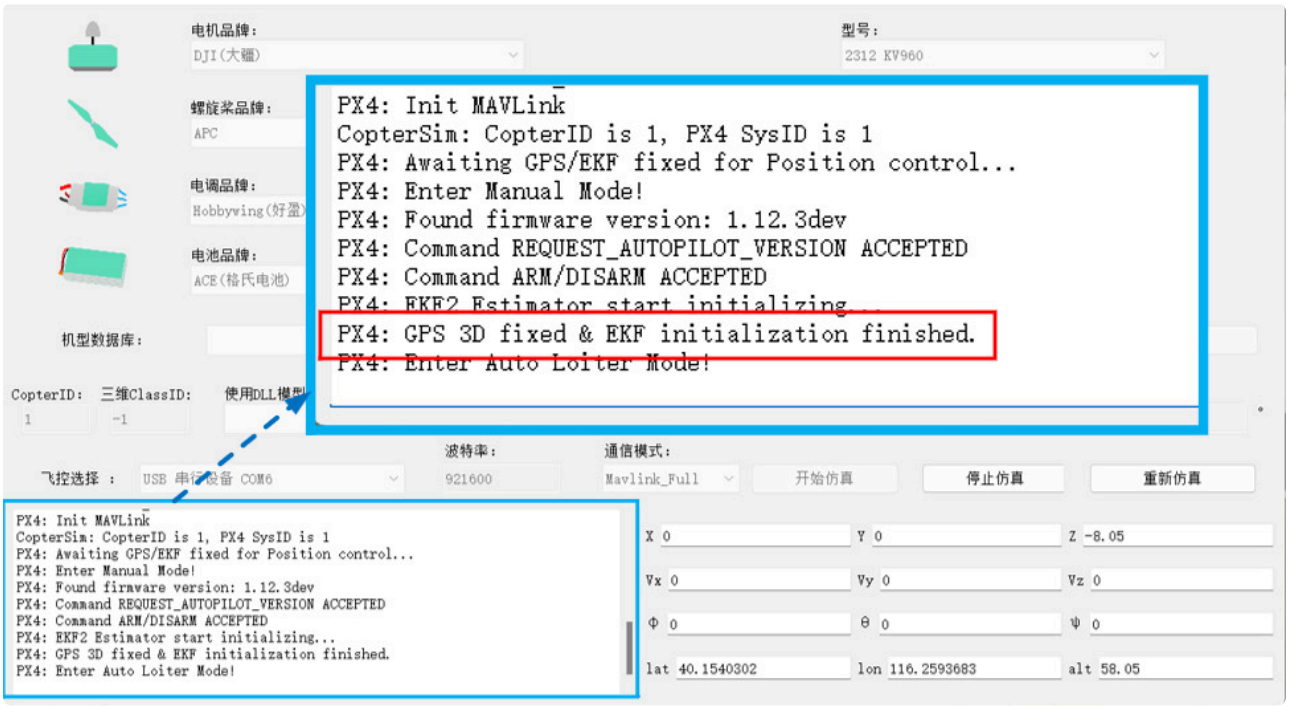
当第3个点，点击完成后，程序将自动计算并规划出路径，如下图所示。



并将生成的二维栅格地图 `planned_path.npy` 与OGM2ESDF.py放在同一个目录下。

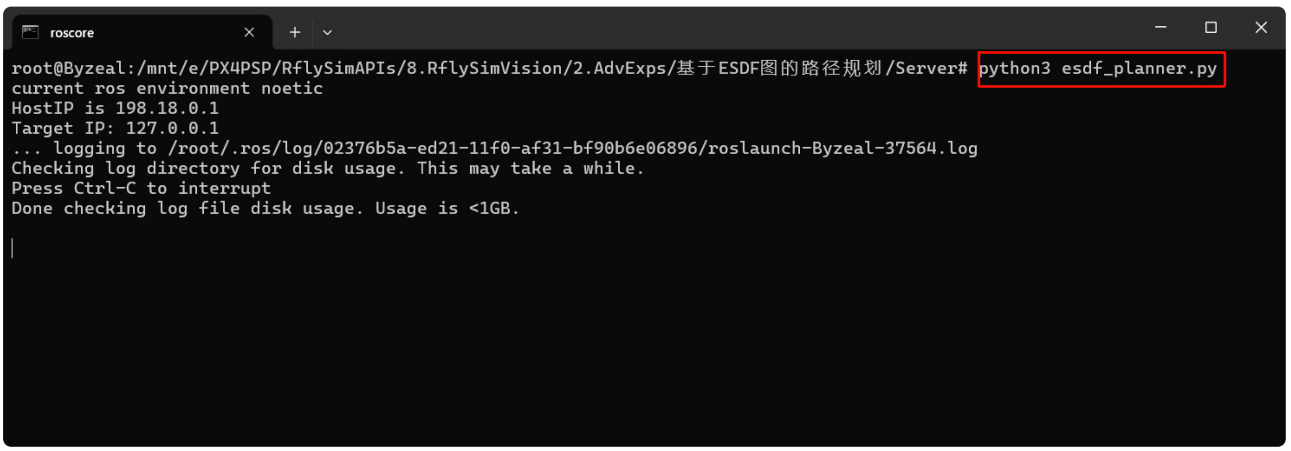
4.2 仿真初始化

1、双击运行`Client\VisionCapAPIDemo.bat`脚本文件，等待仿真环境初始化完成。脚本将会启动 1 个 QGC 地面站， 1 个 CopterSim 、1 个 RflySim3D 软件， 等待CopterSim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。如下图所示：



4.3 ESDF转换和路径规划

复制在4.1小节中生成的`maps\planned_path.npy`文件到`Server`文件夹中，双击运行`Server\WinWSL.bat`脚本文件，在弹出的对话框中输入`python3 esdf_planner.py`。该程序将运行起来。



同时，可以看到RflySim3D中的无人机正常起飞并开始按照步骤4.1中规划的轨迹飞行。

5. 关键知识点

5.1 实验整体思路和框架

本实验通过处理二维栅格地图，构建包含障碍物距离信息的ESDF场和拓扑骨架的Voronoi图，基于这两者进行路径规划，并将规划轨迹映射到RflySim平台中，控制无人机实现闭环避障运动。整体流程如下：

1. **地图处理**：加载二维栅格地图，进行二值化处理，区分障碍物和自由区域。
2. **ESDF转换**：计算欧氏符号距离场，得到每个点到最近障碍物的距离信息。
3. **Voronoi图生成**：提取自由区域的中轴作为拓扑骨架，形成Voronoi图。
4. **路径规划**：在Voronoi骨架上搜索初始路径，然后使用梯度优化算法（类似CHOMP）进行平滑和避障优化。
5. **坐标转换**：将像素坐标的路径转换为物理世界的NED坐标系。
6. **仿真验证**：在RflySim平台中控制无人机沿规划路径飞行，实现闭环避障。

5.2 关键程序解析

1) 栅格地图的转化（OGM2ESDF.py -> MapProcessor类）

```
# 图像二值化处理，阈值为250，阈值以下部分视为障碍物，阈值以上为自由区域
mask_obstacle = self.raw_map_data < self.obs_thresh
self.processed_map_data = np.where(mask_obstacle, 0, 255)
```

```
# 计算自由区域至障碍物的欧氏距离
dist_out = distance_transform_edt(self.mask_free, sampling=resolution)
# 计算障碍物内部至边界的欧氏距离
dist_in = distance_transform_edt(mask_obstacle, sampling=resolution)
# 障碍物内部是负数，外部是正数，边界是0
esdf_map = dist_out - dist_in
```

```
# 获取自由区域的中线作为Voronoi图的骨架
voronoi_skeleton = medial_axis(self.mask_free)
```

2) 起点和终点的选取（OGM2ESDF.py -> PointSelector类）

```
# 弹出原有二维栅格地图
plt.imshow(self.raw_map, cmap='gray', origin='upper')
```

```
# 交互式选取三个点：起点，目标点1，目标点2
points_raw = plt.ginput(n=3, timeout=-1, show_clicks=True)
```

3) 路径规划算法（OGM2ESDF.py -> GradientPathPlanner类）

```
def _skeleton_search(self, start, goal):
    ...
    """使用八邻域搜索算法搜索Voronoi图骨架上的点，找到从起点到目标点的路径"""
```

```
def _optimize_segment(self, start, goal, segment_idx):
    # 对Voronoi图上搜索出来的点进行插值
    path = self._resample_path(raw_path, num_points=150)
    # 计算前进方向的平滑梯度
    smooth_grad[1:-1] = path[1:-1] - 0.5 * (path[:-2] + path[2:])
    # 计算距离障碍物的梯度
    obs_grad[j, 0] = -factor * self.grad_y[py, px]
    obs_grad[j, 1] = -factor * self.grad_x[py, px]
    # 优化函数：路径点沿负梯度方向更新
    path[1:-1] -= alpha * (w_smooth * smooth_grad[1:-1] + w_obs * obs_grad[1:-1])
```

4) 像素空间到物理空间映射（esdf_planner.py -> load_and_process_path函数）

```
# 图片起点为左上角，无人机坐标系为NED，故转换公式为：
# NED_East (y) = (Pixel_Row_Start - Pixel_Row_Current) * Resolution
# NED_North(x) = (Pixel_Col_Current - Pixel_Col_Start) * Resolution
d_row = p[0] - start_point[0]
d_col = p[1] - start_point[1]
pos_y_east =  d_row * resolution
pos_x_north = d_col * resolution
```

5) 无人机控制（esdf_planner.py -> SearchPoint函数）

```
def SearchPoint(speed, rate, PosE, AngEuler, mav, PointList):
    # 计算目标点相对于当前位置的角度
    Angle = math.atan2(Posy - PosE[1], Posx - PosE[0]) * 180 / math.pi
    # 旋转无人机朝向目标点
    # 移动无人机到目标点
    mav.SendPosNED(target[0], target[1], target[2], AngEuler[2])
```

5.3 关键技术原理

1) 欧氏符号距离场（ESDF）

- **距离变换**：对于地图中的任意一点 $p = (x, y)$，其到最近障碍物集合 $\mathcal{O}$ 的欧氏距离定义为：

$$dist(p, \mathcal{O}) = \min_{o \in \mathcal{O}} \|p - o\|_2$$

- **符号距离场**：通过两次距离变换构建ESDF：

$$ESDF(p) = d_{out}(p) - d_{in}(p)$$

其中 $d_{out}(p)$ 是自由空间中的点到最近障碍物的距离，$d_{in}(p)$ 是障碍物内部的点到最近自由边界的距离。

- **物理意义**：$ESDF(p) > 0$ 表示点 p 处于安全区域，数值越大越安全；$ESDF(p) < 0$ 表示点 p 处于障碍物内部；$ESDF(p) = 0$ 表示点 p 处于障碍物边界。

2) Voronoi图与中轴变换

- **Voronoi拓扑路径搜索**：利用中轴变换提取自由空间的骨架。骨架上的点满足到至少两个最近障碍物的距离相等。
- **拓扑优势**：Voronoi骨架提供了自由空间的拓扑结构，能够保证路径在几何中心，最大化与障碍物的距离。

3) CHOMP算法优化路径

- **代价函数**：$J(\xi) = w_s J_{smooth}(\xi) + w_c J_{collision}(\xi)$
  - 平滑代价 J_{smooth}：减少路径长度并保持平滑
  - 避障代价 $J_{collision}$：利用ESDF产生的排斥势场
- **梯度下降**：路径点按照负梯度方向进行迭代更新：

$$q_i^{(k+1)} = q_i^{(k)} - \alpha \left(w_s \frac{\partial J_{smooth}}{\partial q_i} + w_c \frac{\partial J_{collision}}{\partial q_i} \right)$$

6.参考资料

1. **RflySim官方文档**： https://rflysim.com/doc/zh/
2. **Oleynikova, H., et al.** "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning." IROS 2017. 论文链接
3. **Zucker, M., et al.** "CHOMP: Covariant Hamiltonian optimization for motion planning." IJRR 2013. 论文链接
4. **RflySim工具链安装指南**： https://rflysim.com/doc/zh/HowToInstall.pdf
5. **单线激光雷达建图实验**： `[安装目录]\8.RflySimVision\2.AdvExps\e6_LaserSLAMdemo\Readme.pdf`

7.常见问题

Q1：运行OGM2ESDF.py程序时，点击地图选点后程序无响应或报错

现象：双击运行`maps\WinWSL.bat`后输入`python3 OGM2ESDF.py`，程序弹出地图窗口，但点击三个点后程序卡住或报错，无法生成规划路径。

原因：

1. 地图文件路径不正确，程序无法读取`map_cropped.yaml`或对应的PGM地图文件。
2. Python依赖包未安装完整，缺少必要的科学计算库（如scipy、skimage等）。
3. 地图分辨率与程序中的参数不匹配。
4. 选择的点位于障碍物内部或过于靠近障碍物边界。

解决方案：

1. 检查`maps`文件夹中是否存在`map_cropped.yaml`和`map_cropped.pgm`文件。如果不存在，可以运行`auto_crop_map.py`脚本生成裁剪后的地图。
2. 安装所有必需的Python依赖包：在WSL终端中进入`maps`文件夹，运行`pip install -r requirements.txt`。
3. 确保地图分辨率与程序中的参数一致。打开`map_cropped.yaml`文件查看`resolution`值，应与OGM2ESDF.py中的默认值匹配。
4. 选择点时确保点在自由区域（地图白色部分），避免选择障碍物区域（黑色部分）。

Q2：仿真初始化时CopterSim软件未显示"GPS 3D fixed & EKF initialization finished"字样

现象：双击运行`Client\VisionCapAPIDemo.bat`后，CopterSim软件启动，但下侧日志栏一直未显示初始化完成信息，无人机无法正常起飞。

原因：

1. RflySim工具链未正确安装或配置。
2. 网络连接问题导致CopterSim与QGC地面站通信失败。
3. 防火墙或安全软件阻止了相关端口的通信。

解决方案：

1. 确保已按照RflySim官方文档正确安装工具链，并完成基本配置。
2. 检查网络连接，确保所有软件在同一网络环境中运行。
3. 暂时关闭防火墙或安全软件，或添加相关端口的例外规则（如UDP端口14550、20100等）。
4. 重启所有软件，按照正确顺序启动：先启动QGC地面站，再启动CopterSim，最后启动RflySim3D。

Q3：无人机在RflySim3D中不按规划路径飞行或飞行轨迹偏差大

现象：成功运行`esdf_planner.py`后，无人机起飞但飞行轨迹与规划路径不一致，或者无人机在原地盘旋不移动。

原因：

1. 坐标转换错误，像素坐标到物理坐标的转换公式不正确。
2. 地图分辨率参数`MAP_RESOLUTION`设置错误。
3. 路径文件`planned_path.npy`未正确复制到`Server`文件夹，或文件内容损坏。
4. 无人机控制频率与路径点更新频率不匹配。

解决方案：

1. 检查`esdf_planner.py`中的`load_and_process_path`函数，确保坐标转换公式正确。公式应为：

```
NED_East (y) = (Pixel_Row_Start - Pixel_Row_Current) * Resolution
NED_North(x) = (Pixel_Col_Current - Pixel_Col_Start) * Resolution
```

2. 确认`MAP_RESOLUTION`参数与地图YAML文件中的`resolution`值一致（通常为0.05）。
3. 确保将`maps\planned_path.npy`文件正确复制到`Server`文件夹中，并检查文件是否完整。
4. 调整`SearchPoint`函数中的`speed`和`rate`参数，使无人机运动更平滑。

Q4：WSL环境无法启动或Python命令不可用

现象：双击`WinWSL.bat`或`WslGUI.bat`后，窗口一闪而过或提示错误，无法在WSL中运行Python程序。

原因：

1. Windows系统未安装WSL（Windows Subsystem for Linux）。
2. WSL中未安装Python3或相关依赖。
3. 批处理文件中的路径配置错误。

解决方案：

1. 确保已启用WSL功能：在PowerShell中以管理员身份运行`wsl --install`。
2. 在WSL中安装Python3：`sudo apt update && sudo apt install python3 python3-pip`。
3. 检查批处理文件内容，确保WSL发行版名称正确（如`Ubuntu`）。
4. 可以尝试手动打开WSL终端，进入项目目录执行命令。

Q5：ROS话题数据未正常发布或接收

现象：运行`esdf_planner.py`后，在ROS环境中未看到相关的图像或IMU话题数据。

原因：

1. `VisionCaptureApi.isEnableRosTrans`未正确设置为`True`。
2. ROS环境未正确配置或未启动`roscore`。
3. 网络配置错误，ROS节点无法通信。

解决方案：

1. 确保`esdf_planner.py`中已设置`VisionCaptureApi.isEnableRosTrans = True`。
2. 在运行程序前，先启动`roscore`：在WSL终端中运行`roscore`。
3. 检查ROS网络配置，确保`ROS_MASTER_URI`和`ROS_IP`环境变量设置正确。
4. 使用`rostopic list`命令检查话题是否正常发布。

---

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