
场景地图布置接口实验原理

1. 文件目录.....	1
2. 总体说明.....	1
地形的组成.....	1
边界划定.....	1
灰度图高程信息.....	1
地形与物体的交互.....	1
静态景物与地形配准.....	1
动态物体与地形的交互.....	1
3. 关键功能的实现.....	2
3.1. 地形文件.....	2
3.1.1. 地形高程信息 png.....	2
3.1.2. 地形配准信息 txt.....	2
3.2. 添加标靶和障碍物并与场景地形匹配.....	2
3.2.1. 获取物体初始高度与场景地形表面的偏移量.....	2
3.2.2. 考虑偏移量重新创建物体.....	2
3.2.3. 自动获取地形偏移量并创建物体.....	2
3.3. Simulink 接口函数.....	3
1. LoadPngData:	3
2. GetTerrainAltData:.....	3
3. RflySendUECMD:	3
4. RflyCameraPosAng:	4
3.4. 生成物体移动轨迹并与场景地形匹配.....	5
3.4.1. 加载地形高程数据 LoadPngData.m.....	5
3.4.2. 读取指定位置的地形高度 getTerrainAltData.m.....	10
3.4.3. 生成贴合地面的移动轨迹 TrajGen.slx.....	11
4. 相关文献.....	11
附加资源.....	12

1. 文件目录

例程目录: [\[安装目录\]\RflySimAPIs\4.RflySimModel\0.ApiExps\14.inCopterData\](#)

文件夹/文件名称	说明
1.TargetCreateKey\Readme.pdf	场景标靶和障碍物布置与地形匹配实验步骤
2.TargetCreatePy\Readme.pdf	
3.TargetPlace\Readme.pdf	
4.GetTerrainMAT\Readme.pdf	场景物体运动轨迹生成与地形匹配实验步骤
5.TrajGen\Readme.pdf	
6.TrajGenMulti\Readme.pdf	
7.HeterTrajGenMulti\Readme.pdf	
8.TenCarCircleCtrl\Readme.pdf	

2. 总体说明

地形的组成

边界划定

灰度图高程信息

计算机以二进制记录数据，所以其量化的等级以二进制来划分，即 2^n 。若用 n 个比特 (bit) 来记录每个像元，则其灰度值范围可在 0 到 2^n-1 之间，如 8-bit 的数据取 $2^8=256$ 灰度级 (其值 0~255)；若规定用 1 比特来记录每个像元，其灰度值仅为 0 和 1，即所谓的二值图像。若用彩色系统来记录图像，根据色度学原理，任何一种彩色均可由红 (R)、绿 (G)、蓝 (B) 三原色按适当比例合成，若用 8 比特的 RGB 彩色坐标系来记录像元，可记录 $2^{24}=16777216$ 种不同的 RGB 组合。其中，若 RGB 的亮度值分别为 0, 0, 0，则产生黑色像元；若 RGB 为 255, 255, 255，则产生白色像元，若 RGB 的亮度值相等，则产生灰度效果。

地形与物体的交互

静态景物与地形配准

动态物体与地形的交互

为了实现动态物体 (如无人机、标靶、障碍物等) 与地形的交互，需要利用碰撞检测技术，判断动态物体是否与地形或其他物体发生接触。主要包括两个任务：粗筛和细化。粗筛是对所有可能发生碰撞的物体进行快速的预处理，排除掉一些明显不会产生碰撞的物体，减少计算量。细化是对粗筛后剩余的物体进行更加精确的碰撞判断，确定碰撞的位置、时间、法向量等信息。

3. 关键功能的实现

3.1. 地形文件

3.1.1. 地形高程信息 png

➤ 详见[1]中的 7.1.2 地形高程信息(“****.png”)

3.1.2. 地形配准信息 txt

平台中 txt 校正文件存储的是右上角三维坐标点 (xy 全为正, z 向上为正)、左下角三维坐标点 (xy 全为负, z 向上为正)、第 3 点三维坐标点, 单位均为厘米。前两个点的目的是为了确认地形的范围和中心坐标, 第 3 点坐标可自行选取, 理论上需要尽量在高度上与前两个点有落差, 用于校正高度尺度。

➤ 详见[1]中的 7.1.3 地形校准数据(“****.txt”)

3.2. 添加标靶和障碍物并与场景地形匹配

3.2.1. 获取物体初始高度与场景地形表面的偏移量

RflySim3D 内置的快捷键交互功能使得鼠标双击能捕捉到场景中的对象, 获取其在世界场景中的位置 (object pos), 以及击中点的位置 (click point)。根据获取的击中点的位置信息, 可以得到模型与场景地形间的偏差。

3.2.2. 考虑偏移量重新创建物体

在模型对应的 XML 配置文件中为模型初始位置添加相应的偏移量, 即可使之后在相同位置创建的模型初始高度与地形匹配。

同理, 也可以在 python 脚本中为模型创建位置添加相应的偏移量, 可以调整在场景中创建的模型的位置以使其与地形匹配。示例如下:

```
ue.sendUE4Cmd('RflyChangeMapbyName Grasslands', 0)
```

调用 RflySim3D 控制台命令'RflyChangeMapbyName Grasslands'修改 UE 场景。这里的 RflyChangeMapbyName 命令表示切换地图(场景), 后面的字符串是地图名称, 这里会将所有打开的窗口切换为草地地图。sendUE4Cmd 函数在 UE4CtrlAPI.py 库文件中的完整定义

```
sendUE4Cmd(cmd, windowid ==-1)
```

其中 cmd 为命令字符串, windowid 为接收窗口号(假设同时打开多个 RflySim3D 窗口), windowid ==-1 表示发送到所有窗口。

```
ue.sendUE4Pos(1001,200150,0,[43.943,1.620,-11.465],[0,0,0])
ue.sendUE4Pos(1002,100150,0,[72.946,-4.388,-11.465],[0,0,0])
ue.sendUE4Pos(1003,151,0,[115.915,5.510,-10.448],[0,0,0])
```

在指定初始位置, 以指定姿态创建指定样式的物体

3.2.3. 自动获取地形偏移量并创建物体

使用 sendUE4Pos 接口创建物体, 需要详细地设定物体的 xyz 坐标, 因此如果想要物体完全贴在 (站在) 地面上, 就需要去测量地表的高度 (这里通过在 RflySim3D 中双击地面

想要放置靶标的位置，就能在 Click Point 中读出 z 的坐标)。

除了 sendUE4Pos 函数，我们还提供了一个自动确定地面高度，将物体挪至地表的接口函数 sendUE4Pos2Ground

```
ue.sendUE4Cmd('RflyChangeMapbyName MountainTerrain')
```

切换地图到山地场景

```
ue.sendUE4Pos(100,30,0,[ -16.479,1.45,-177.181],[0,0,math.pi])
```

根据在地图场景中测量得到的地形高度，创建人物的初始位置

```
ue.sendUE4Pos2Ground (101,30,0,[ -17.479,1.45,0],[0,0,math.pi])
```

使用该命令会自动调用地形服务接口 UEMapServe.py，测量场景地形高度并在对应位置创建人物模型

3.3. Simulink 接口函数

➤ 详见[1]中的 [Simulink 接口函数](#)

RflySim3D 关于 MATLAB 的外部接口与 Python 相类似，都是通过收发 UDP 消息与 RflySim3D 进行交互，这里主要包括如下几类命令：

1. LoadPngData:

生成地图高度数据，会生成一个“MapHeightData.mat”文件来存储高度图矩阵数据。该接口会在“CopterSim\external\map”中读取对应地形的 png 高度图。该 png 高度图是通过 RflySim3D 对地形的扫描得到的（用[1]中的控制台命令 RflyScanTerrainH 可以扫描得到场景的高度图）。

函数解释见[加载地形高程数据 LoadPngData.m](#)

2. GetTerrainAltData:

输入地图的 x, y 坐标，输出当前地形高度 z。该函数根据上面生成的 MapHeightData.mat 获取地形的高度。

函数解释见[读取指定位置的地形高度 getTerrainAltData.m](#)

3. RflySendUECMD:

向 RflySim3D 发送控制台命令，这些命令与[1]Python 接口中介绍的控制台命令是完全一样的，只不过一个是用 Python 发的，一个是用 Simulink 发的，发出来的 UDP 结构是一样的。

以下是对函数 RflySendUE4CMD 的解释：

```
function RflySendUE4CMD(x)
```

% 这个函数向 RflySim3D 发送一个控制台命令，和 Python 方法发送的 UDP 结构是相同的。

% x y z 是相机的位置，单位是米，使用北东地坐标系

% roll pitch yaw 是相机的方向，单位是度

```
if ~exist('x','var')
x='RflyChangeMapbyName Grasslands';
end
```

% 如果没有给出输入参数 `x`，就使用一个默认的命令来切换地图到草原

```
out = uint8(x);
```

% 把命令字符串转换成一个 `uint8` 数组

```
len=length(out);
```

% 获取数组的长度

```
yy=[out,uint8(zeros(1,52-len))];
```

% 用零来填充数组，使它的长度为 52 字节

```
yy=[typecast(int32(1234567890),'uint8'),yy];
```

% 在数组的前面加上四个字节的头部，这是 1234567890 的 `int32` 值的 `uint8` 格式

```
u=udp('255.255.255.255','RemotePort',20010);
```

% 创建一个 UDP 对象，使用广播地址和端口号

```
fopen(u);
```

% 打开 UDP 连接

```
fwrite(u,yy);
```

% 把数据写入 UDP 连接

```
fclose(u);
```

% 关闭 UDP 连接

```
delete(u)
```

```
end
```

% 删除 UDP 对象

4. RflyCameraPosAng:

设置相机的位置

```
function RflyCameraPosAng(x,y,z,roll,pitch,yaw)
```

这段代码定义了一个名为 `RflyCameraPosAng` 的函数，该函数接受六个参数：`x`，`y`，`z`，`roll`，`pitch` 和 `yaw`。

- `x`，`y` 和 `z` 是相机位置在北东地（NED）坐标系中的坐标，单位是米。NED 坐标系是一个固定在地球上的参考系，其 `x` 轴指向北方，`y` 轴指向东方，`z` 轴指向下方。

- `roll`，`pitch` 和 `yaw` 是相机方向的角度，单位是度。`roll` 是绕 `x` 轴的旋转，`pitch` 是绕 `y` 轴的旋转，`yaw` 是绕 `z` 轴的旋转。相机方向是由 Tait-Bryan 角度定义的，它们指定了如何从 NED 坐标系旋转到相机坐标系。

```
if ~exist('x','var')
    x=0;
end
if ~exist('y','var')
    y=0;
end
if ~exist('z','var')
    z=0;
end
if ~exist('roll','var')
    roll=0;
end
if ~exist('pitch','var')
    pitch=0;
```

```

end
if ~exist('yaw','var')
    yaw=0;
end

```

- 函数首先检查是否有任何参数缺失，如果有，就将它们赋予默认值零。这意味着函数可以用少于六个参数来调用，而缺失的参数会被假定为零。

```

out = uint8(['RflyCameraPosAng ',num2str(x),' ',num2str(y),' ',num2str(z),' ',num2str(roll),' ',num2str(pitch),' ',num2str(yaw)]);

```

- 函数然后创建一个字符串，该字符串包含了函数的名称和六个参数，用空格隔开。例如，如果 $x = 1$, $y = 2$, $z = 3$, $roll = 10$, $pitch = 20$, $yaw = 30$ ，那么字符串就是 'RflyCameraPosAng 1 2 3 10 20 30'。

```

len=length(out);
yy=[out,uint8(zeros(1,52-len))];

```

- 函数将字符串转换为 8 位无符号整数的向量，这样可以作为字节在网络上传输。函数在向量后面补零，直到它的长度为 52，这是 UDP 消息的固定长度。

```

yy=[typecast(int32(1234567890),'uint8'),yy];

```

- 函数在向量前面加上四个字节的头部，这是整数 1234567890 转换为 8 位无符号整数的结果。这个头部用于标识消息为 RflySim3D 的命令，RflySim3D 是一个软件，它可以在 3D 环境中模拟无人机和其他车辆的飞行。

```

u=udp('255.255.255.255','RemotePort',20010);
fopen(u);
fwrite(u,yy);
fclose(u);
delete(u)
end

```

- 函数创建一个 UDP 对象，该对象可以将数据发送到广播地址 255.255.255.255 的 20010 端口。这意味着消息会被发送到同一网络上监听该端口的所有设备。函数打开 UDP 对象，将字节向量写入它，关闭它，并删除它。

```

%u3=udp('127.0.0.1','RemotePort',8848,'LocalPort',8850);%同上

```

- 还可以创建另一个 UDP 对象，该对象可以将数据发送到本地地址 127.0.0.1 的 8848 端口，并在 8850 端口接收数据。这部分代码被注释掉了，也就是说它不会被执行。这可以用于测试或调试的目的，以在同一台机器上发送和接收消息。

3.4. 生成物体移动轨迹并与场景地形匹配

3.4.1. 加载地形高程数据 LoadPngData.m

```

function LoadPngData(pngMapName)

```

LoadPngData 是一个 matlab 函数，它接受一个可选的输入参数 pngMapName，这是一个包含地形高程数据的 PNG 文件的名称。这个函数尝试找到并读取这个文件，并返回一个归一化的灰度值矩阵，以及图像的尺寸。

根据输入地图名确定要检索的地形文件

```

if ~exist('pngMapName','var')
    pngMapName='MountainTerrain';

```

end

首先检查是否提供了输入参数 pngMapName。如果没有，它使用一个默认值 'MountainTerrain'。

检索地形文件

```
filelocPath='map';  
fileLocPng = [filelocPath, '\', pngMapName, '.png'];
```

然后构造文件路径，把文件名和'.png'扩展名拼接到一个文件夹名'map'后面。

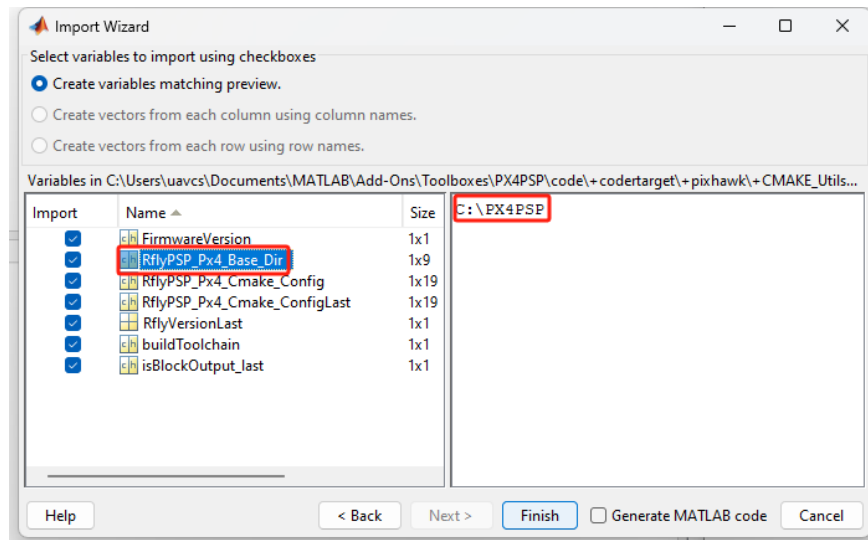
该函数会依次在当前目录下的 map 文件夹、“..\..\CopterSim\external\map”文件夹、“【平台安装路径】\CopterSim\external\map”文件夹，这三个路径下搜索目标名的png与txt。具体判断逻辑如下：

首先检索 png 文件：

```
if ~exist(fileLocPng, 'file')  
    filelocPath='..\..\CopterSim\external\map';  
    fileLocPng = [filelocPath, '\', pngMapName, '.png'];  
    if ~exist(fileLocPng, 'file')  
        matPath=[userpath, '\Add-  
Ons\Toolboxes\PX4PSP\code\codertarget\+pixhawk\+CMAKE_Uutils\FirmwareVersion.mat'];  
        if exist(matPath, 'file')  
            AA=load(matPath);  
            filelocPath=[AA.RflyPSP_Px4_Base_Dir, '\CopterSim\external\map'];  
            fileLocPng = [filelocPath, '\', pngMapName, '.png'];  
            if ~exist(fileLocPng, 'file')  
                error(['Cannot find file: ', pngMapName, '.png file']);  
            end  
        else  
            error(['Cannot find file: ', pngMapName, '.png file']);  
        end  
    end  
end
```

1.使用 exist 函数检查文件是否存在于当前目录。如果存在，代码跳过后续的搜索过程，直接进入文件读取部分。如果不存在，代码进入第一个 if 语句的内部，尝试寻找文件的备用位置。在第一个 if 语句的内部，代码构造了一个新的文件路径，存储在变量 filelocPath 中。这个文件路径是从当前目录往上三层，然后进入 CopterSim\external\map 子文件夹。这个文件夹是 CopterSim 软件存储其外部文件，如地图和模型的地方。将 filelocPath 和 fileLocPng 拼接起来，得到一个新的文件路径，覆盖原来的 fileLocPng。

2.然后，再次使用 exist 函数检查文件是否存在于这个新的路径下。如果存在，代码跳出第一个 if 语句，进入文件读取部分。如果不存在，进入第二个 if 语句的内部，尝试另一种方法。在第二个 if 语句的内部，寻找一个名为 FirmwareVersion.mat 的文件，这个文件位于用户路径（通常是 C:\Users\用户名\Documents\MATLAB\）的附加工具箱文件夹中 PX4PSP 代码子文件夹下。这个 mat 文件包含了 RflySim 平台安装的基本目录。



3.将这个 mat 文件的路径存储在变量 `matPath` 中，并使用 `exist` 函数检查文件是否存在。如果文件存在，代码使用 `load` 函数加载这个 mat 文件，并将其中的内容存储在一个结构体变量 `AA` 中。然后，代码从这个结构体中提取一个字段，名为 `RflyPSP_Px4_Base_Dir`，它表示 RflySim 平台的安装路径。代码将这个路径和 `CopterSim\external\map` 子文件夹以及文件名拼接起来，得到一个新的文件路径，覆盖原来的 `fileLocPng`。然后，代码再次使用 `exist` 函数检查文件是否存在于这个新的路径下。如果存在，代码跳出第二个 `if` 语句，进入文件读取部分。如果不存在，代码抛出一个错误信息，指示 PNG 文件找不到，并终止程序的运行。

在相同路径检索 txt 文件：

```
fileLocTxt = [fileLocPath, '\', pngMapName, '.txt'];
```

现在，函数已经确定了 PNG 文件的位置，并将其存储在 `fileLocPng` 中。函数还需要一个 TXT 文件，它和 PNG 文件同名，但扩展名不同。这个 TXT 文件包含了一些元数据，如 PNG 文件的宽度、高度、分辨率等。函数使用类似的方法，将 `fileLocPath` 和 `pngMapName` 以及 `.txt` 拼接起来，得到一个 TXT 文件的路径，存储在变量 `fileLocTxt` 中。

```
if ~exist(fileLocTxt, 'file')
    error(['Cannot find file: ', pngMapName, '.txt']);
end
```

使用 `exist` 函数检查文件是否存在，如果不存在，函数抛出一个错误信息，指示 TXT 文件找不到，并终止程序的运行。

读取地形文件

首先读取 txt 文件：

```
fileID = fopen(fileLocTxt);
```

如果 TXT 文件存在，函数使用 `fopen` 函数打开文件，返回一个文件标识符，存储在变量 `fileID` 中。

文件标识符是一个整数，表示文件的状态和位置。

```
m_readData_cell = textscan(fileID, '%f', 9, 'Delimiter', ',');
```

函数使用 `textscan` 函数读取文件的内容，返回一个单元格数组，存储在变量

`m_readData_cell` 中。

`textscan` 函数的参数说明如下：

- `fileID`: 文件标识符，表示要读取的文件。
- `%f`: 格式字符串，表示要读取的数据类型。`%f` 表示浮点数。
- `9`: 数字，表示要读取的数据的数量。这里是 TXT 文件中的前 9 个数据。
- `'Delimiter'`: 字符串，表示数据之间的分隔符。这里是逗号 (,)，因为 TXT 文件是用逗号分隔的。

单元格数组是一种数据类型，它可以包含不同类型和大小的数据，每个单元格用花括号访问。

```
m_readData = m_readData_cell{1};
```

`m_readData_cell` 是一个单元格数组，只包含一个单元格。这个单元格的内容是一个 9x1 的矩阵，存储了 TXT 文件中的数据。函数使用花括号提取这个矩阵，并赋给一个变量 `m_readData`。

```
[m,n]=size(m_readData);  
if m~=9 || n~=1  
    error(['Cannot parse data in ',fileLocTxt]);  
end
```

函数使用 `size` 函数获取 `m_readData` 这个矩阵的行数和列数，存储在变量 `m` 和 `n` 中。如果 `m` 不等于 9 或者 `n` 不等于 1，说明 TXT 文件的格式有误，函数抛出一个错误信息，指示 TXT 文件无法解析，并终止程序的运行。否则，函数继续执行。

读取 png 文件

```
rowmap = imread(fileLocPng);  
rowmap = double(rowmap)-32768;  
[rows, columns] = size(rowmap);
```

函数接着使用 `imread` 函数读取 PNG 文件，这个函数返回一个灰度图像矩阵。函数将这个矩阵转换为双精度浮点数，并减去 32768，得到一个在 -32768 和 32767 之间的整数值矩阵。这些值表示地形的高程，值越高，对应的海拔越高。函数使用 `size` 函数获取 `rowmap` 这个矩阵的行数和列数，存储在变量 `rows` 和 `columns` 中。

校准数据

```
PosScaleX = (m_readData(1)-m_readData(4))/(columns-1);  
PosScaleY = (m_readData(2)-m_readData(5))/(rows-1);
```

% 根据第 1 点（区域左上角）和第 2 点（区域右下角）的 x 坐标之差与列数之比计算 x 轴的水平缩放因子

% 根据第 1 点和第 2 点的 y 坐标之差与行数之比计算 y 轴的水平缩放因子

```
PosOffsetX = m_readData(4);  
PosOffsetY = m_readData(5);
```

% 将第 2 点的 x 坐标赋值为 x 偏移量

% 将第 2 点的 y 坐标赋值为 y 偏移量

```
intCol = int32((m_readData(7)-PosOffsetX)/PosScaleX + 1);
intRow = int32((m_readData(8)-PosOffsetY)/PosScaleY + 1);
```

% 将第 3 点的 x 坐标转换为 png 文件中对应的列索引

% 将第 3 点的 y 坐标转换为 png 文件中对应的行索引

```
heightInit = double(rowmap(1,1));
heightFirst = double>(rowmap(rows,columns));
heightThird = double(rowmap(intRow,intCol));
```

% 将 png 文件左上角的高程值赋值为初始高度

% 将 png 文件右下角的高程值赋值为第 2 点的高度

% 将第 3 点对应的行列索引处的高程值赋值为第 3 点的高度

```
if abs(heightThird-heightFirst)<=abs(heightThird-heightInit)
    if abs((heightInit-heightThird))>10
        PosScaleZ = (m_readData(6)-m_readData(9))/((heightInit-heightThird));
    else
        PosScaleZ = 1;
    end
else
    if abs(heightThird-heightFirst)>10
        PosScaleZ = (m_readData(3)-m_readData(9))/((heightFirst-heightThird));
    else
        PosScaleZ = 1;
    end
end
```

% 检查第 2 点和第 3 点中哪一个与初始高度的高度差更大

% 如果第 3 点的高度差更大，检查高度差是否大于 10

% 如果是，根据第 1 点和第 3 点的 z 坐标之差与高度差之比计算 z 轴的垂直缩放因子

% 如果不是，将垂直缩放因子设为 1

% 如果第 3 点的高度差更大，检查高度差是否大于 10

% 如果是，根据第 2 点和第 3 点的 z 坐标之差与高度差之比计算 z 轴的垂直缩放因子

% 如果不是，将垂直缩放因子设为 1

```
intPosInitZ = heightInit;
```

% 将初始高度赋值为初始 z 位置

```
PosOffsetZ = m_readData(6);
```

% 将第 2 点的 z 坐标赋值为 z 偏移量

```
xMax=abs(m_readData(1)/100);
```

% 根据第 1 点的 x 坐标的绝对值除以 100 计算 x 轴的最大值

```
yMax=abs(m_readData(2)/100);
```

% 根据第 1 点的 y 坐标的绝对值除以 100 计算 y 轴的最大值

```
binmap= -(PosOffsetZ + ((rowmap)-intPosInitZ)*PosScaleZ)/100.0;
```

% 将高程值转换为实际的 z 值，方法是将 z 偏移量、垂直缩放因子和除以 100 的操作应用于高程值

```
save('MapHeightData', 'binmap', 'PosOffsetX', 'PosScaleX', 'PosOffsetY', 'PosScaleY');
```

% 将变量 binmap, PosOffsetX, PosScaleX, PosOffsetY, PosScaleY 保存到一个名为

MapHeightData 的文件中

3.4.2. 读取指定位置的地形高度 getTerrainAltData.m

```
function zz = getTerrainAltData(xin,yin)
```

该函数的功能是读取指定位置的地形高度，输入参数是 `xin` 和 `yin`，分别表示 `x` 方向和 `y` 方向的坐标（单位是米），输出参数是 `zz`，表示对应的地形高度（单位也是米）。

```
mapname='MapHeightData';
persistent binmap;
persistent PosOffsetX;
persistent PosScaleX;
persistent PosOffsetY;
persistent PosScaleY;
if isempty(binmap)
    SS=load([mapname,'.mat']);
    binmap=SS.binmap;
    PosOffsetX=SS.PosOffsetX;
    PosScaleX=SS.PosScaleX;
    PosOffsetY=SS.PosOffsetY;
    PosScaleY=SS.PosScaleY;
end
```

函数首先检查是否已经加载了地形数据，如果没有，则调用 `load` 函数，从 `MapHeightData.mat` 文件中读取变量 `binmap`、`PosOffsetX`、`PosScaleX`、`PosOffsetY`、`PosScaleY`。这些变量分别表示转换后的位置数据、`x` 方向上的偏移量、`x` 方向上的缩放比例、`y` 方向上的偏移量、`y` 方向上的缩放比例。这些变量使用 `persistent` 关键字声明，表示它们会在函数的多次调用之间保持不变。

```
intCol = (xin*100-PosOffsetX)/PosScaleX+1;
intRow = (yin*100-PosOffsetY)/PosScaleY+1;
intColInt=floor(intCol);
intRowInt = floor(intRow);
a=intCol-intColInt;
b=intRow-intRowInt;
intRowInt1=intRowInt+1;
intColInt1=intColInt+1;
```

然后根据输入的坐标 `xin` 和 `yin`，计算出它们对应的矩阵索引 `intCol` 和 `intRow`。这里需要将坐标乘以 100（因为 UE 中的单位是厘米，这里需要换算为米），然后减去偏移量，再除以缩放比例，最后加 1。因为 Matlab 的矩阵索引是从 1 开始的，而坐标是从 0 开始的，所以要加 1。

```
[m,n]=size(binmap);
if intColInt<1
    intColInt=1;
    intColInt1=1;
    a=0;
end
if intColInt>=n
    intColInt=n;
    intColInt1=intColInt;
    a=0;
end
if intRowInt<1
```

```

intRowInt=1;
intColInt=1;
b=0;
end
if intRowInt>=m
    intRowInt=m;
    intColInt=intColInt;
    b=0;
end

```

接着对矩阵索引进行取整和边界检查，确保它们不超过矩阵的范围，也不小于 1。如果有越界的情况，就将索引设为边界值，并将插值系数 a 或 b 设为 0。

```

zz=binmap(intRowInt,intColInt)*(1-b)*(1-a)+binmap(intRowInt1,intColInt)*b*(1-a)+binmap(intRowInt,intColInt1)*(1-b)*a+binmap(intRowInt1,intColInt1)*b*a;

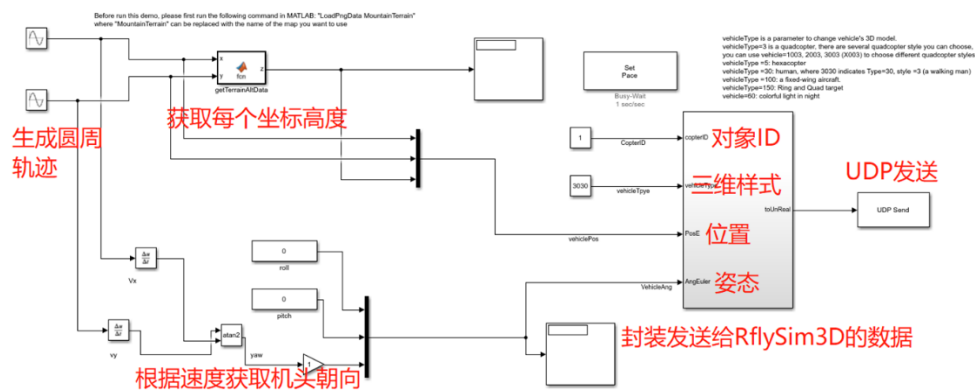
```

最后使用双线性插值的方法，根据索引周围的四个点的高度值，计算出指定位置的高度值 zz。双线性插值的公式是：

$$zz = f(x,y) = f(0,0)*(1-x)*(1-y) + f(0,1)*x*(1-y) + f(1,0)*(1-x)*y + f(1,1)*x*y$$

其中 x 和 y 是小数部分，f(i,j)是整数部分对应的高度值。这样就可以得到一个平滑的地形高度，而不是直接采用最近的整数索引的高度值。

3.4.3. 生成贴合地面的移动轨迹 TrajGen.slx



首先调用 LoadPngData.m 函数，加载对应场景的地形数据（txt 校准数据和 png 高程灰度图）生成对应的地形高度矩阵 mapheightdata.mat。然后在 simulink 模块中，输入模型的运动轨迹，先根据位置调用 getTerrainAltData.m 函数解析地形高度矩阵 mapheightdata.mat 以实时解算出模型所处位置的地形高度，接着根据速度和角速度解算出模型的姿态。将得到的位置和姿态封装成 udp 包，连同模型的 id 和三维样式一起发送给 RflySim3D，RflySim3D 收到消息后就会创建对应的贴合地形的运动模型。

之后将这种模块复制多份，设置延时启动，即可得到多个模型的运动轨迹。只要更改输入的轨迹、模型 id 和三维样式，即可得到不同模型的运动轨迹。只要为各个模型的初始位置添加间隔，并使用相同的形状输入作为轨迹，就可以构成编队。

4. 相关文献

- [1]. [..\API.pdf](#)
- [2]. [IConsoleManager | Unreal Engine Documentation](#)

附加资源

官方文档: RflySim 官方文档: <https://rflysim.com/doc/zh/>

社区交流: 加入 RflySim 技术交流群: 951534390

