多无人机路径规划说明文档

18373263 贺梓淇

代码结构

本代码共有四个文件，按照继承顺序如下：

- `GlobalVars.py` ：保存无人机、威胁、奖励的参数以及代码中的可调参数
- `Utils.py` ：包含了计算 `reward map, threat map, danger map` 以及保存图片结果的工具函数
- `UAVControls` ：包含两个类 `UAV` 以及 `UAVGroup` ，分别用于控制单个无人机以及控制无人机组
- `main.py` ：运行方法示例

原理

本份代码的基本原理与所给的单无人机导航基本相同

通过将每个无人机视为一个威胁，每次移动后更新路径规划即可达到无人机之间互相避障的目的。由于每次移动更新之后威胁矩阵会随着无人机位置的变化更新，所以可以每次规划只规划一步来减少计算量

另一方面，加入了奖励机制使得无人机组可以完成更加复杂的动作

在此之上，我对部分细节算法进行了一些修改，可以让无人机更好、更安全地完成任务

改进点

01：防止无人机因为过高的诱惑飞入障碍物

如果简单粗暴地在 `weight map` 上减去 `reward map` 的值，会有在本有危险的地方被 `reward` 减为负数，从而引诱无人机飞向该处的风险

为了避免这种情况的出现，我认为有必要根据 `threat map` 的值对 `reward map` 做出修正，修正方式如下：

$$R_{modified} = \frac{R_{original}}{e^{TH*factor}}$$

其中 TH 是威胁系数图，$factor$ 是一个可调参数，对应代码中的 `TR_COEFFICIENT`

02：优化无人机相对奖励的飞行路线

在测试中出现过无人机直接飞向奖励值较高的奖励区，而忽略了身边奖励值较低的奖励区的问题，为了解决这个问题，我使用了贪心的方法根据无人机的当前位置调整 `reward map` 的值，对于每一点修正方式如下：

$$R'_{(x,y)} = R_{(x,y)} * (1 - ||(x,y) - (x_{uav}, y_{uav})||)^{factor}$$

这样可以降低与无人机距离较远的奖励区的权重，`factor` 是一个可调参数，对应代码中的 `REWARD_POINT_DECAY`

03：针对无人机和威胁的不同调整

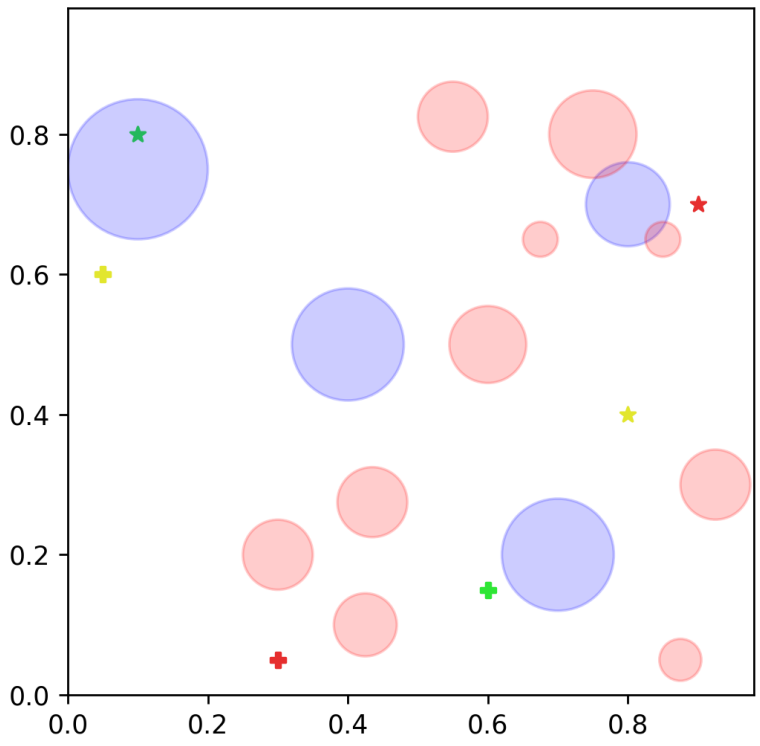
对于一般的威胁而言，使用高斯函数计算分布并不一定是一种最安全的做法。无人机仍然有可能飞入威胁范围之中。因此，我修改了无人机与威胁的距离的计算方法。新的计算方法将计算无人机与威胁圈的距离，而不是无人机与威胁中心的距离

另一方面，对于视为质点无人机之间而言，并没有必要引入威胁圈。因此我对不同类型的威胁系数的计算方法做了简单的区分

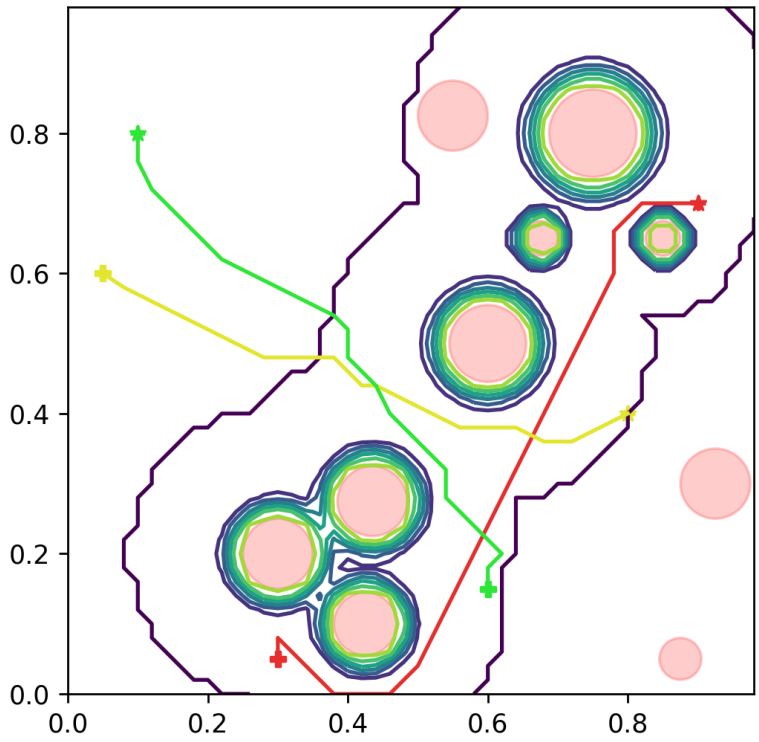
运行结果

见 `results` 文件夹

每个文件夹中保存的 `.py` 文件即为生成改结果的代码



起始情况



27步迭代后