智能无人机技术设计实践

--实验4指导书

李溢

联系方式: liyi16@mails.tsinghua.edu.cn

时间：2019.10.26

目 录

- 1. 安装python库
- 2. A*算法和Dijkstra算法实现
- 3. GUI查看(可选)

1 安装Python库

- 安装python包

```
>>> pip install numpy
>>> pip install matplotlib
```

2 A*算法实现

提示：所有资料在网络学堂的实验4资料文件夹下。

① 初始化

- 加载地图 map.npy 文件，非0值表示障碍物
- 指定起点和终点
- 确定可移动的方向

```
##############################################
########     initialization       ########
##############################################
#mapmap = np.load('map.npy')
mapmap = np.zeros((15,15),dtype=np.int)
print(mapmap.shape)
startPosition = (2, 0)     #Initial point
goalPosition = (13, 11)    #End point
direction = [(0,1),(0,-1),(1,0),(-1,0),(1,1),(1,-1),(-1,1),(-1,-1)]
mapRow, mapCol = mapmap.shape;

# visualization
mapViz = mapmap.copy()
mapViz[startPosition[0],startPosition[1]] = 20;
mapViz[goalPosition[0], goalPosition[1]] = 30;
plt.pcolormesh(mapViz)
plt.show()
```

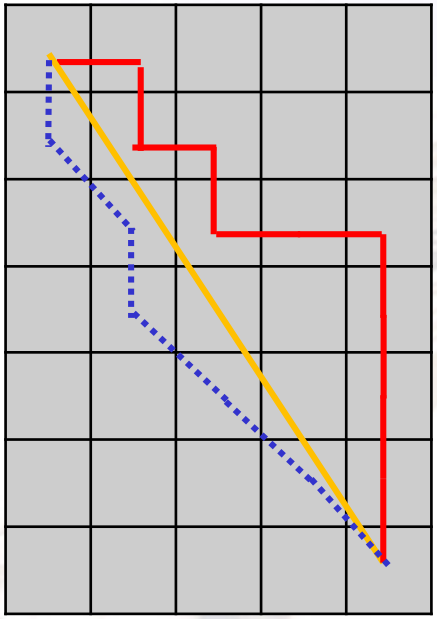
2 A*算法实现

② 定义代价函数和启发式函数

- A*算法中：f=g+h
- Dijikstra算法中只考虑 g

③ 定义任意其他需要的函数

```
##########################################
########    heuristic function    ########
##########################################
def g(parameters):
    # code here ...
    pass

def h(parameters):
    # code here ...
    pass

def f(parameters):
    # code here ...
    pass
```



曼哈顿距离

```
dx=abs(startPosition[0]-goalPosition[0])
dy=abs(startPosition[1]-goalPosition[1])
cost = dx+dy
```

对角距离

```
dx=abs(startPosition[0]-goalPosition[0])
dy=abs(startPosition[1]-goalPosition[1])
cost = dx+dy-0.6*min(dx,dy)
```

欧几里得距离

```
dx=abs(startPosition[0]-goalPosition[0])
dy=abs(startPosition[1]-goalPosition[1])
cost = sqrt(dx^2+dy^2)
```

2 A*算法实现

④ A*算法核心部分

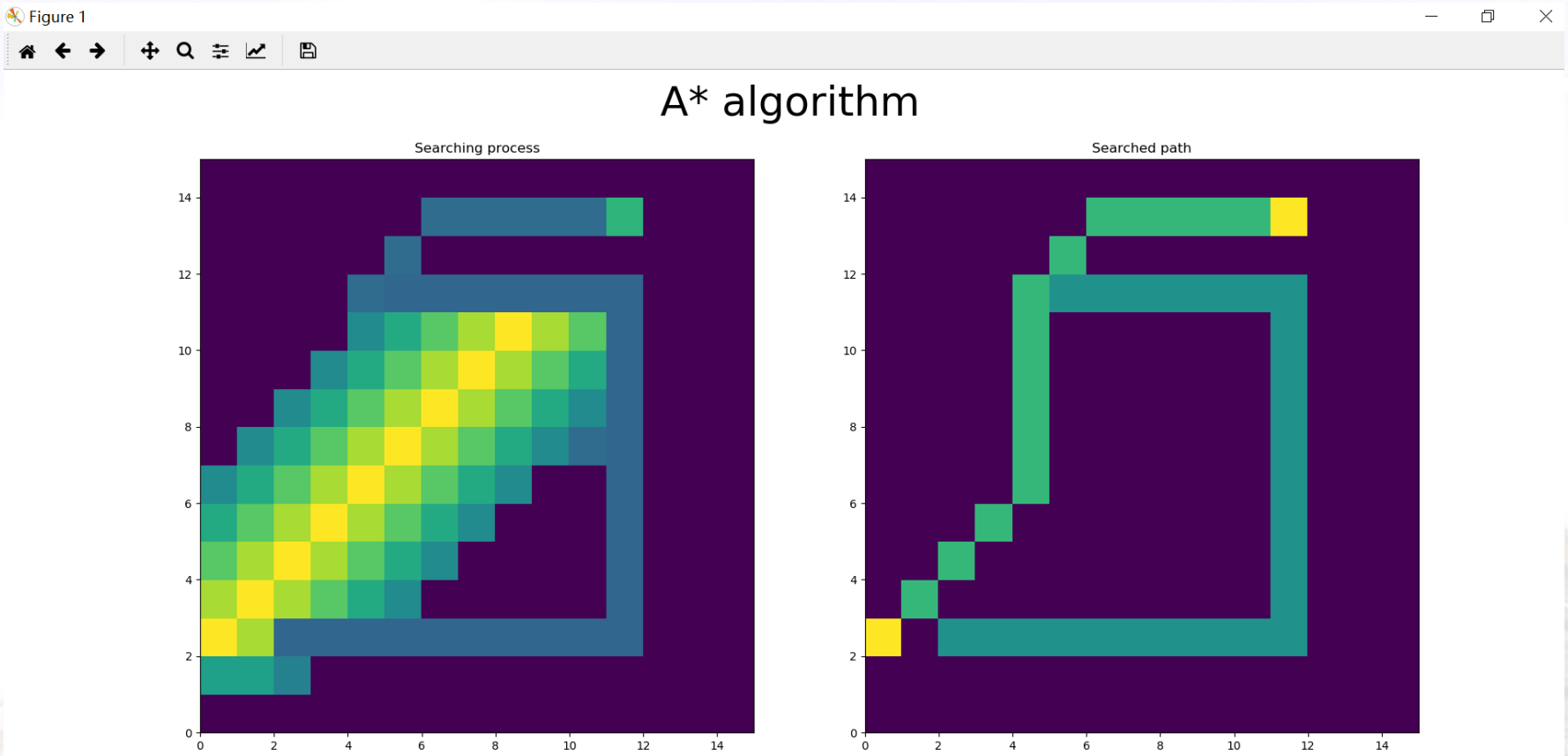
- 维护 openList 和 closeList
- 期望的格式：
  - openList 和 closelist 分别是一个 list
  - List 中的每个元素都是一个tuple(x, y, f, g, h, parnetNodeIdx)
  - X,y 分别为当前点的横纵坐标
  - F,g,h分别为A*算法中的三个函数输出值(Dij算法中f,h可以是任意值)
  - parnetNodeIdx为当前节点在closelist中父节点的序号，“父节点”的含义参考A*算法
- 上面格式要求非强制，但是改动后可视化部分也需要对应修改

```
#####################################
########       A* algorithm       ########
#####################################
# Advertised format of elements in openList and closeList is
# tuple(x,y,f,g,h,parnetNodeIdx)
# x: row of current node
# y: column of current node
# f,g,h: f,g,h in A* algorithm, f=g+h
# parentNodeIdx: index of current node's parent node in closeList
# ---------------------------------------
# You can change these, but remember code for visualization
# should also be modefied.

# initialization
openList = []
closeList = []

while True:
    ###################################
    ########     code here ...    ########
    break
    ###################################
```

2 A*算法实现

⑤ 可视化(助教已完成)

- 可视化需要用到前面得到的 closelist，注意格式，或自行修改
- 左侧(动图)表示路径搜寻过程，右侧表示寻找到的最优路径



3 GUI查看

- 本部分为可选实验，可提供交互式界面查看路径规划算法的正确性
- 安装依赖
  - Pip install pythotk

① 下载并进入 GUI 文件夹

② 封装

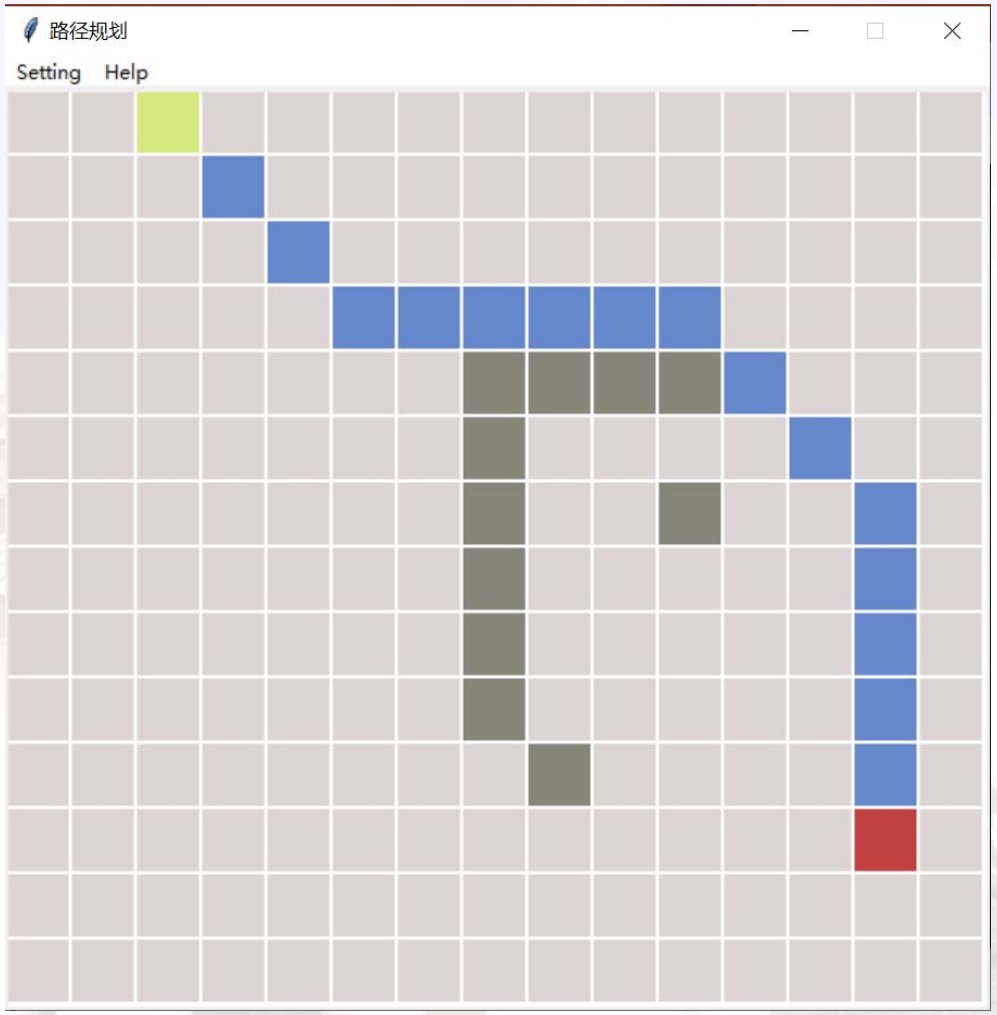
- 将前面完成的路径规划算法封装为函数 findPath, 写入 findPath.py 文件
- findPath函数要求如下(可见showPath_GUI.py文件)

```
你只需要在同一目录下findPath.py文件中实现以下函数：
-----------------------------------------------------
def findPath(map, startPosition, goalPosition):
    ...
    return path

map: 一个矩阵，'1'元素表示有障碍物，'0'表示没有
startPosition: 一个tuple，存放起点(x,y)坐标
goalPosition: 一个tuple，存放终点(x,y)坐标
return: 一个list，每个元素表示路径上的一个点，例如
    [(1,2),(1,3),(2,4)]
-----------------------------------------------------
findPath函数中可以采用任意路径规划算法，只需要按照
上述格式返回计算得到的路径，比如你可以对 Astar_demo
中的代码稍加修改，将其封装为findPath()函数
```

3 GUI查看

④ 运行showPath_GUI.py文件

- 单机左键：设置/取消障碍物
- 双击左键：改变起点位置
- 单机右键：改变终点位置



作业提交

◆ 任务要求

修改“Astar_demo.py”，实现A*算法。满足：

① 使用“map.npy”作为测试地图；

② 保留closeList，其他变量可自行设计；

③ 搜索方向 direction 已经定义好，可以改变搜索顺序；

④ 启发式距离函数可以自定义；

⑤ 保留可视化结果截图，请将文件名改为“result_yourstudentID”；

⑥ 最后打包提交自己的源代码与“result_yourstudentID”图片文件。

⑦ 源代码请**详细地**写出注释，也可以另写一份实验报告做说明。

谢谢！

答疑地点：罗姆楼11层101房间