消火栓&灭火器校核开发文档

1.背景介绍

本算法目标为，对于给定的空间结构以及对应的消火栓以及灭火器位置，得到满足特定需求的覆盖区域，以提示空间中未被保护的区域。其中消火栓的覆盖范围由水龙带长度与喷射出的水柱长度决定，而灭火器的覆盖范围由保护距离决定。

geojson文件的输入包括空间（space）、障碍（obstruct）、门（door）、平面灭火器即消火栓、手提灭火器即灭火器。这些由多边形表示，通过多边形的邻接与包含关系即可得到具体的空间结构与连通关系。

在实际空间中可能存在单向门以及门的开闭变化，该算法也应提供对于该方面的控制。

2.算法接口

输入：

1. 表示建筑结构的geojson文件，即多边形与类型;
2. 消火栓的水龙带长度L_0以及水柱长度L_1;
3. 灭火器的的最大保护距离L_2;
4. 参数opt1，决定校核消火栓还是灭火器
5. 参数opt2，设置空间包含时增加的厚度，即墙体厚度
6. 参数opt3，需要校核的空间编号
7. 参数opt4，设置门的性质
8. 参数opt5，设置获取水柱覆盖范围的采样精度
9. 参数opt6，楼层高度设置
10. 其他参数

输出：

geojson文件的空间表示以及覆盖区域，消火栓的单股与双股用不同颜色表示

3.算法思路

3.1 基本思路

1.水龙带的覆盖区域即可达区域的获得：通过从初始的消火栓开始获得可见区域并获得不可见区域与可见区域的分界点重复该过程，直到所有区域均可见或者水龙带长度用完，最终得到覆盖区域。

2.连通区域的获得：对于通过门相连的邻接空间，对于三个多边形进行并操作即可获得。当两个空间具有包含关系且内部空间有门时，对于内部空间依据墙体将其拓展为外部空间的洞，再将门去除以获得连通空间的多边形表示。

3.门性质的控制：以连通区域为点，门为边建立有向图即可表示门的性质。对于每个连通区域，其有效的消火栓为当前连通区域的以及可到达该连通区域的消火栓。

4.楼梯的处理：将其视为一个点，之后通过计算每个消火栓到楼梯口的距离将其转化为对应的消火栓。

3.2 数据结构

```
//有向图类
Edge{
    int DestVertex; //有向边的终点
    int index;      //对应的门的编号
    Edge *next;
}
Vertex{
    int index;      //对应空间的编号
    Edge *adj;      //相邻边的起始边
}
directed_graph{
    vector<Vertex> vertex_array;
    void insertV(int index); //插入顶点
    void insertE(int start, int end, int index);//插入边
    void deleteE(int start, int end, int index);//删除边
    vector<Vertex> getConnected(int start);     //得到从该点开始的所有连通区域
}
//覆盖区域的获得由一个函数表示
get_covered(Polygon_with_holes_2 env, Point_2 p);
//两点距离的计算
get_dist(Polygon_with_holes_2 env, Point_2 p0, Point_2 p1);
//输入结构
inputStruct{
    Polygon_with_holes_2 poly;
    string type;
}
//geojson文件中的每个块的表示
linkblock {
    Polygon_with_holes_2 poly;  //由geojson文件中数据直接得到的多边形
    double index;               //编号
    string type;                //类别
    Point_2 center;             //中心点
    set<int> connect;           //相连多边形标号
    set<int> within;            //被包含多边形编号
    set<int> contain;           //包含的多边形的编号
};
//连通区域表示
connected_components{
    Polygon_with_holes_2 poly;  //连通区域的多边形表示
    double index;               //连通区域编号
    set<int> incp;              //包含的连通区域的编号
    set<int> fh;                //包含的消火栓编号
    set<int> fe;                //包含的灭火器编号
    set<pair<int,int>> prev;    //前向的连通区域与对应门的编号
    set<pair<int,int>> next;    //后向的连通区域与对应门的编号
}
//数据的总体表示
blueprint{
    vector<linkblock> data;     //初始数据
    linkblock outerbound;       //用以处理外部灭火器添加的包含整个空间的矩形区域
    set<int> space;
    set<int> obstruct;
    set<int> door;
    set<int> fh;
    set<int> fe;
```

```
    set<int> stair;              //对应各类的编号集合
    directed_graph g;
    blueprint(inputStruct *indata, Para *p);     //初始化数据与参数
    setdoor(int doorindex, int start, int end, bool sw); 设置门的开闭与方向
    get_graph(); //基于门的开闭以及方向设置得到有向图
    transfer_stair(opt6); //基于graph以及楼层直接距离的设置将楼梯口转化为对应的消火栓
    get_final(opts);       //基于graph对于每个消火栓的连通区域获取覆盖范围并汇总
}
```

目前使用一个有向图的数据结构记录空间的连通关系，并通过对于边的增删对应门的开闭与方向。

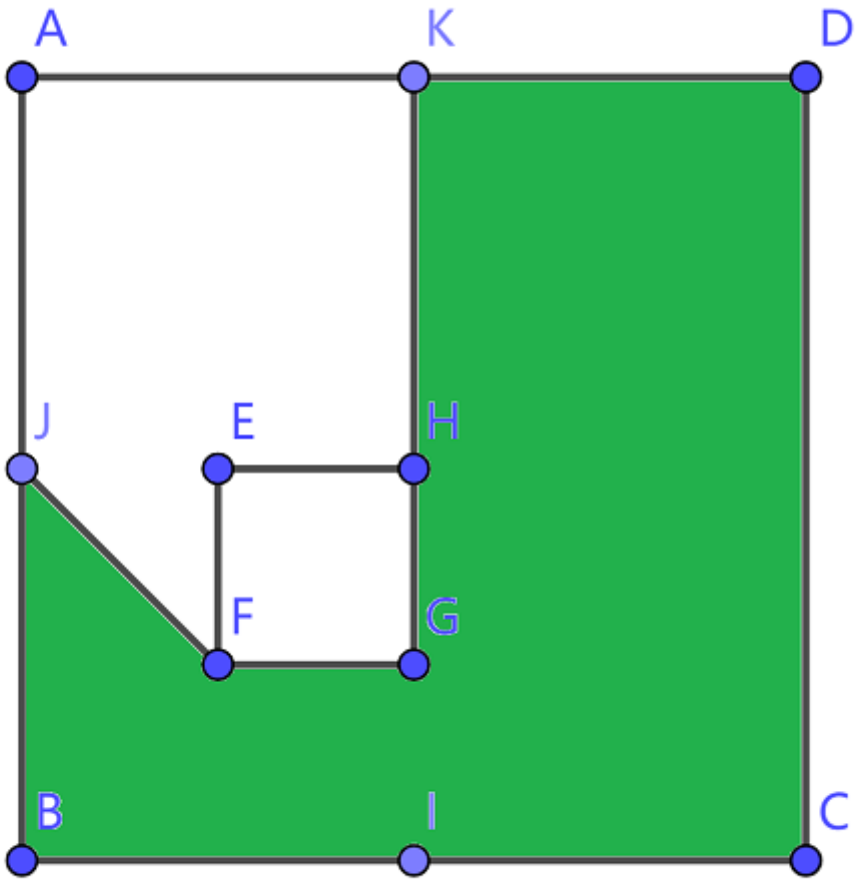
3.3 算法过程

3.3.1 数据处理

1. 对于输入的geojson文件将其解析出多边形以及类型信息，获得多边形之间的关系后存储到linkblock结构中；
2. 对于类别为空间（space）的数据将其内部的障碍物以及其他空间利用多边形的差操作去除，之后获取对于灭火器和消火栓以及楼梯口的包含情况。使用有向图的数据结构记录空间的连通关系，并通过对于边的增删对应门的开闭与方向。

3.3.2 覆盖区域获得

这一部分的基础是可见区域的获得，这里我采用了CGAL中的Triangular_expansion_visibility_2这一方法，覆盖区域获得的基本思路示例如下：



图中多边形为矩形ABCD带有空洞EFGH，I为求覆盖范围的起点。

该算法的输入为多边形poly，多边形内部一点p，以及覆盖距离r；

1. 对于p求其在多边形内部的可见区域，例子中为图中的绿色部分；
2. 可见区域与不可见区域的分隔点，例子中为图中的点F和H；
3. 对与每个分隔点获取其位于哪个不可见区域中，图中F和H均位于AJFEHK组成的多边形中；
4. 对于每个分隔点获取其距离p（图中的I）的距离d，将r-d作为其新的覆盖距离，如果r-d<0，则不考虑该点；
5. 对于p，其可见区域的覆盖范围即为以p为圆心r为半径的圆与可见区域的交。
6. 对于分隔点、其对应不可见区域以及新的覆盖距离重复步骤1到5直到没有新的分隔点出现。
7. 将步骤5中得到的所有多边形求并即为最终的覆盖区域。

对于水柱，由于其只能走直线，只需要考虑可见区域的覆盖范围，对于单个点，获得可见区域并求与以该点为圆心水柱长度为半径的圆求交即可获得。

在得到上述水龙带的覆盖区域之后，水柱的覆盖区域由对于边缘采样之后对于每个点分别求可见区域中的覆盖范围并求并即可获得。

距离计算的思路类似，将循环的终止条件改为终点位于可见区域内，并在循环的过程中记录当前距离并最终求最小值即可。

3.3.3 最终覆盖范围的获得

对于每个消火栓，首先基于有向图获取其连通区域，之后在该区域中计算覆盖范围，最终将所有消火栓得到的结果综合即可得到最终总体覆盖范围。

4.开发环境

编程语言：C++
编程环境：Visual Studio 2019
所需软件包：CGAL5.2，GLUT

2021年3月29日