防雷暗敷引下线算法开发文档

1 背景介绍

在建筑设计过程中，必然要考虑到防雷系统（Lightning Protection System）的设计。当前的防雷系统主要包含接闪带、接闪网、引下线和接地网。在防雷系统当中，**引下线系统**（Down Conductor System）是一个重要的组成部分，它负责将电从楼高处的接闪带逐层引导到地下。引下线分为明敷和暗敷两类，明敷引下线主要应用在屋顶突出构件上，线路直接暴露在外表；暗敷引下线主要应用在楼层之间的传输上，布置位点取于楼内的竖向构件当中，无法直接从楼房外表看到。

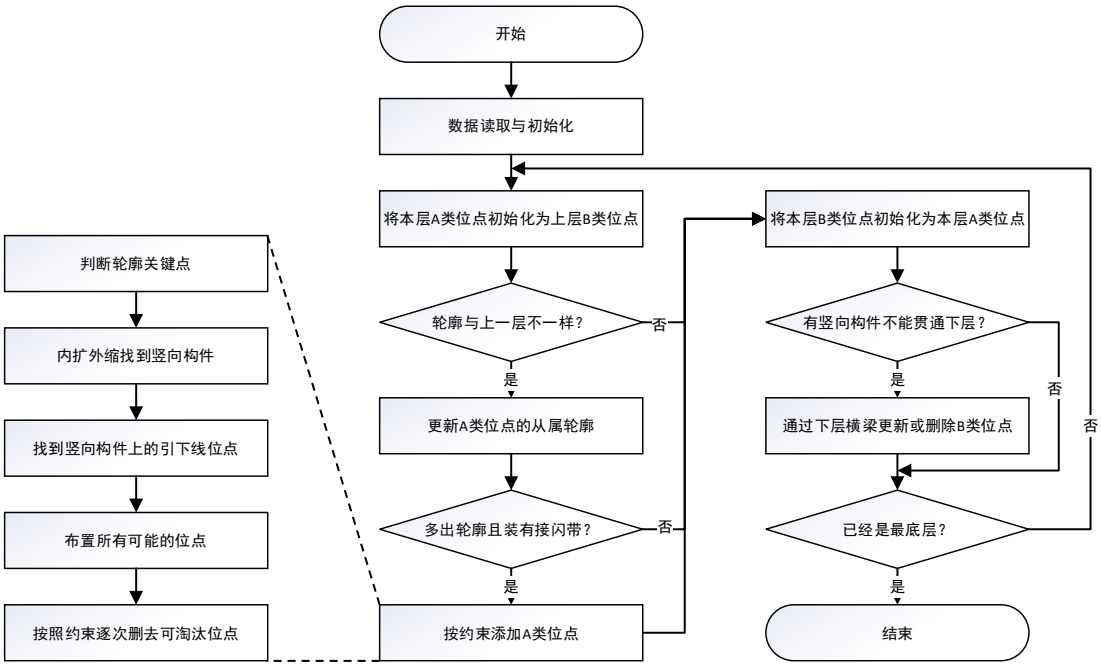
本算法的目标是：根据自大屋面而下各楼层的平面图纸，解析出每一层的建筑外轮廓、竖向构件和横梁，实现建筑单体各楼层（暗敷）引下线的自动布置。

2 算法思路

2.1 核心思想

对于每个楼层求出所有可能的位点。首先对这些位点都布置上引下线，再根据相关约束逐步过滤掉可以不布置的位点。

2.2 流程图



算法从顶层开始布置。对于每一层来说，引下线可以分为两类：一类从本层天花板引到本层地面，记为 A 类；一类从本层地面引到下层，记为 B 类。布置时先 A 类后 B 类。

如果本层轮廓（含内外轮廓，下同）与上层轮廓完全相同，则本层 A 类可以直接照抄上层 B 类；否则引下线位点对应的轮廓与投影点都可能发生变化，需要及时更新。如果本层轮廓较上层轮廓多出的部分（可能多出多个不连通的部分）装有接闪带，则还需要按照约束添加 A 类位点。

添加 A 类位点是本项目的核心部分。首先找到本楼层轮廓的关键点备用；然后用内扩外缩的方法找到位于轮廓附近的所有竖向构件；对于每个竖向构件，找到可以布置引下线的位点；将找到的所有位点都布置上引下线，并维护其对应的引下线 id、从属的轮廓列表（可能不止一个）及分别对应的投影边、投影点；最后按照需求中的约束条件，逐点检查可否删去，对于删去后整体仍满足约束条件的点则舍弃掉。

接下来布置本层 B 类位点。如果本层 A 类位点所在竖向构件能直接贯通到下一层，则可照搬其坐标；否则需要找到下一层中与本层竖向构件通过横梁相连的所有竖向构件，按照其到外轮廓的距离升序查找，若找到了未布置引下线的竖向构件，就将引下线移动到它里面去；若找不到未布置引下线的竖向构件，则舍弃该点，该点对应的引下线也中断于此。

对于每一层数据都执行上述操作，一直到最后一层。最后的结果是每一层的 A 类位点列表和 B 类位点列表，且每个位点除了基本的坐标信息以外，还包括对应的引下线 id。在本项目当中，这两个列表会以 geojson 的格式呈现出来。

2.3 判断轮廓的关键点

根据需求，外轮廓的关键点应满足以下三点要求：

①对于一个多边形轮廓的顶点，它的起始边、顶点、终边需要构成逆时针转动关系；

②它的起始边和终边的边长均大于某一给定阈值；

③在轮廓所在二维平面里，轮廓可以与一给定半径的圆外接于该顶点。

判断的难点在于③。本项目给出的解决方案是：**第一步先求出给定半径的圆沿着外轮廓滚动一周时圆心的运动轨迹，第二步再判断该顶点是否对该运动轨迹产生了贡献，若有贡献则说明③成立**。其中第一步的本质就是对外轮廓做圆头外扩，这一操作 CGAL 并无直接支持，需要用平面图形的集合运算手动求出；第二步的本质就是将外扩后的区域减去以该顶点为圆心、给定半径的圆形区域，只取其外轮廓检查是否较减去之前发生了变化，若有变化则说明有共享，即③成立。

对于内轮廓的某顶点，只需要满足上述条件②时就算作关键点。

2.4 内扩外缩找到竖向构件

基于尽量采用现有模块的理念，本项目拟优先采用 CGAL 当中 2D Straight Skeleton and Polygon Offsetting 模块提供的 create_interior_skeleton_and_offset_polygons_with_holes_2()函数来完成带洞多边形的内扩外缩操作。如果后续测试发现耗时太长，或者业务需求不能使用尖头的内扩外缩，则可以考虑改为检测竖向构件到楼层轮廓的最短距离，并检验该距离是否小于某阈值。如果按照改后的方案，CGAL 提供的多边形之间的距离函数仅适用于凸多边形，和本项目的需求多数情况不符合，所以需要手动实现。

2.5 找到竖向构件里的引下线位点

本项目将所有竖向构件分为三类：

①总共四条边，且最长边不长于最短边的两倍；（视为柱）

②总共四条边的其他竖向构件；（视为无拐点的墙）

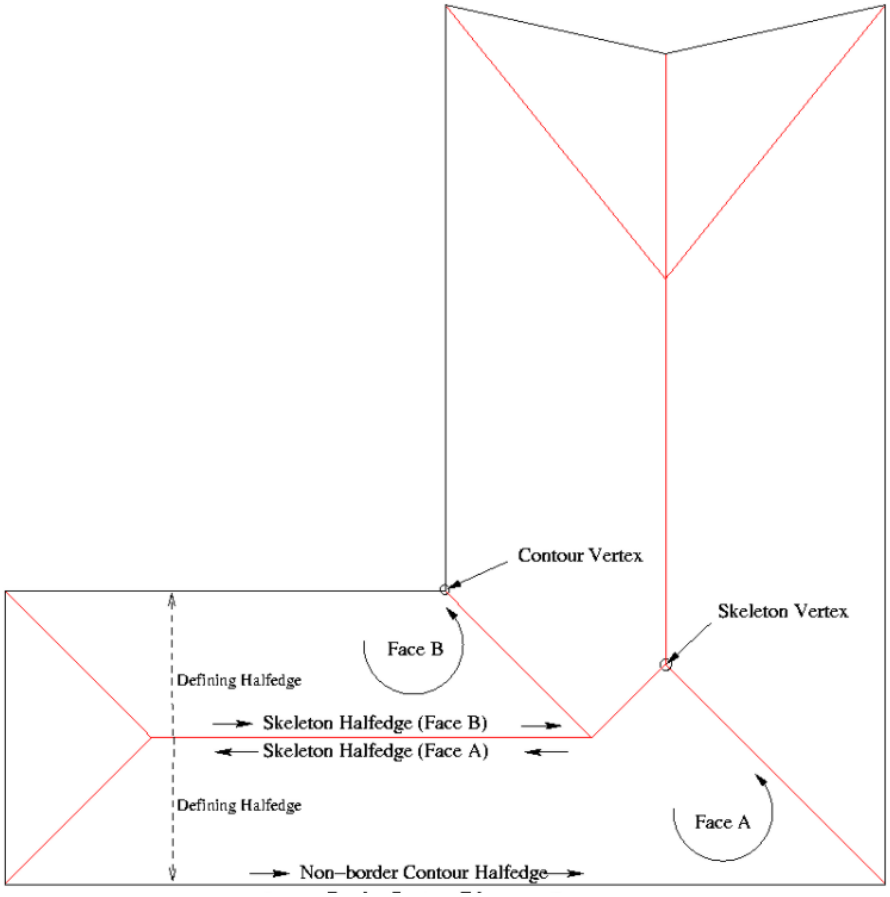
③其他竖向构件。（视为有拐点的墙）

对于①，其潜在引下线位点只可能位于四个顶点的平均点处；对于②和③，统一采用 CGAL 提供的 2D Straight Skeleton and Polygon Offsetting 模块来处理。具体地，对于一个竖

向构件（本质上是不带洞的多边形），利用 create_interior_straight_skeleton_2()函数找到它的骨架（straight skeleton），然后找到骨架当中的骨架顶点（skeleton vertex），示意图见下。

特殊地，对于③，我们只想要拐点不想要端点，而直接求骨架顶点的话也包含了端点。为了筛掉这些端点，可以对该点附近**完全在多边形内部的**半边（half edge）计数，对于端点该总数一定等于 1，其余拐点该总数一定大于 1，这样就可以区分开了。

按照上述方式计算，一个竖向构件可能会对应多个潜在位点。我们选取距离楼层外轮廓最近的点作为该竖向构件最终的引下线安装位点。也就是说，如果该竖向构件要布置引下线，那么只有可能布置在此处。



2.6 按照约束舍弃引下线位点

首先把所有可以布置的位点都布置上，并分轮廓按投影边点的顺序分别组织各位点数据。然后将以下位点上锁（即不允许被筛掉）：

①从上层 B 类引下线照搬过来的位点；

②本层新增的，且存在轮廓上一个关键点与之构成最短距离的位点。

然后遍历位点列表中尚未上锁的点并逐一做判定，若同时满足以下条件：

①在该点所从属的轮廓上（可能有多条轮廓），该点以左和以右两个点的“沿轮廓距离”小于 18 米

②在删除该点后，该点所从属的轮廓上（可能有多条轮廓）各点间平均“沿轮廓距离”小于 18 米；

③全局位点数仍不小于 10；

④该点所在的新增区域内，剩余位点数仍不小于 2。

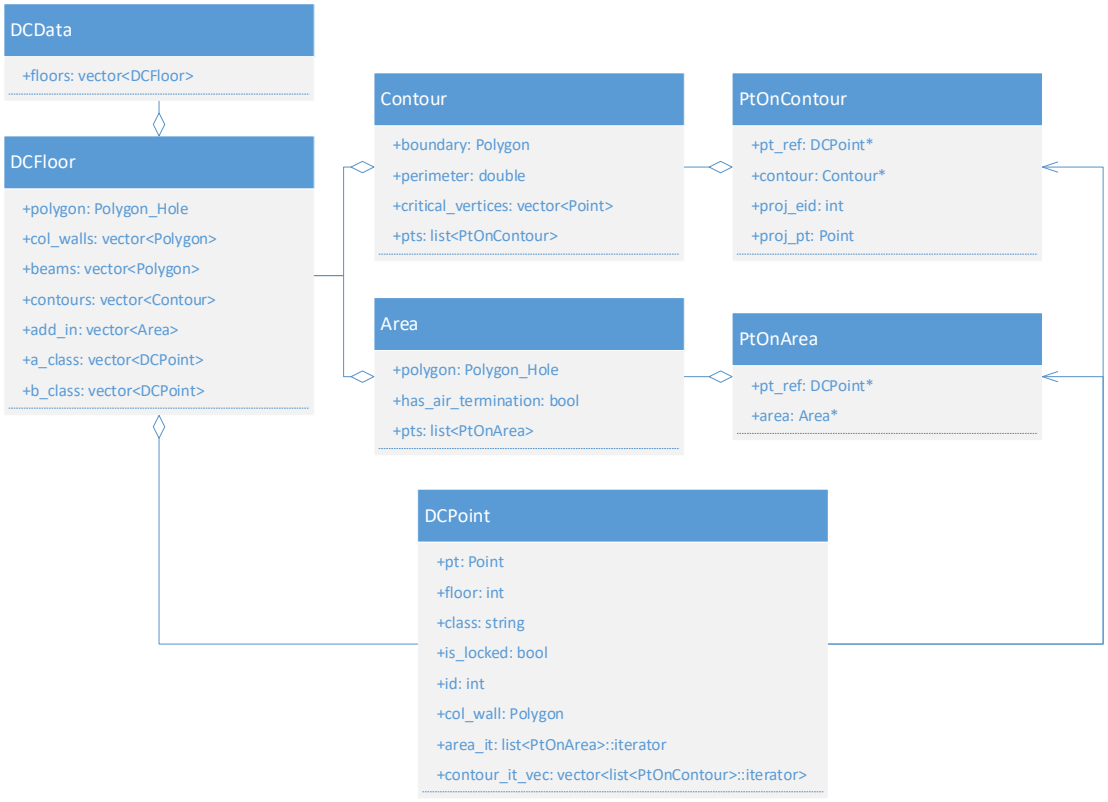
则可以将该点删去。重复上述过程，直到找不出可以删去的点为止。

2.7 基于横梁的竖向构件搜索

通过 CGAL 提供的 2D Regularized Boolean Set-Operations 模块，实现竖向构件是否贯通到下一层的判定、竖向构件与下一层横梁的连接性搜索，以及下一层横梁与下一层竖向构件的连接性搜索。

3 数据结构与算法接口

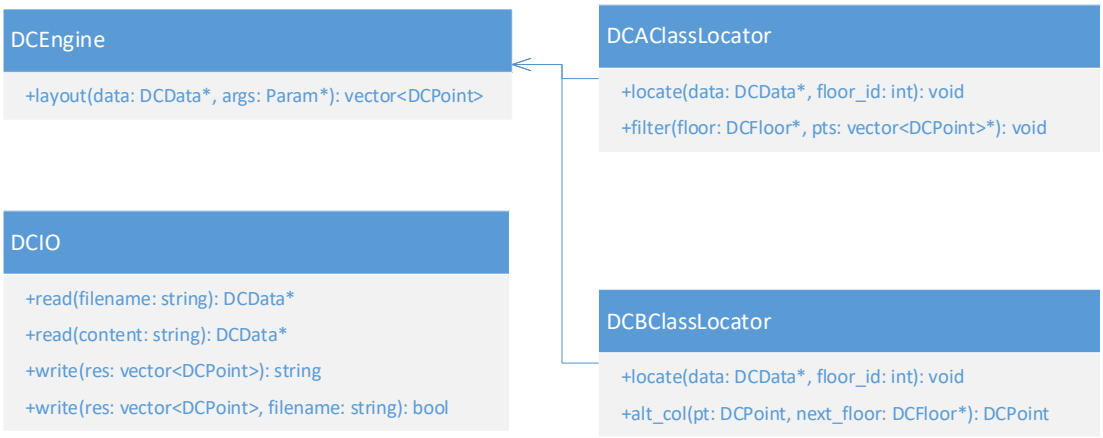
该项目采用以下一些类或结构来存储空间数据。DCData 是最顶层的数据结构，也是算法类调用数据时接收的数据类型。DCFloor 封装了单个楼层的必要信息，包含楼层内外轮廓、该层内墙柱与横梁，以及按照区域和轮廓组织的引下线位点列表信息。Contour 封装了单个轮廓的重要信息，包括轮廓本身、周长、关键点列表以及从属于该轮廓的所有引下线列表。PtOnContour 封装了一个引下线位点从所属轮廓的角度需要保存的信息，例如它对应的 Contour 对象指针、投影到该轮廓的边 id 和点坐标。Area 则封装了一个区域的重要信息（本层轮廓比上一层轮廓多出的一个连通部分称为一个区域，相邻楼层之间可能会出现零个、一个或多个区域），包含区域对应的多边形、是否装有接闪带以及该区域内的所有引下线位点列表。PtOnArea 封装了一个引下线位点从所在区域的角度需要保存的信息。例如它对应的 Area 对象指针。最终一个基本的引下线位点用 DCPoint 类去表示，里面包含了位点坐标、所属楼层与类别（A 类还是 B 类）、引下线 id、是否上锁、所属竖向构件等关键信息。



PtOnContour 和 PtOnArea 必须依赖于 DCPoint 而存在，可以看作一个 DCPoint 在不同的视角下的延伸，这样就实现了一个引下线位点基本属性和特有属性的分离。PtOnContour

和 PtOnArea 可以通过指针找到原本的 DCPoint 对象；DCPoint 也可以通过迭代器直接访问它对应的 PtOnContour 和 PtOnArea 对象。

该项目的算法实现部分也用类进行了封装，实现了与数据分离的目的，并且按照算法的不同步骤和部分拆成若干的子模块，尽量满足了高内聚和低耦合。以下是本项目用到的所有算法类的类图。



为了将原始数据封装成 DCData 以供算法处理，本项目设计了 IO 类 DCIO，该类负责调用 read 方法从文件或者字符流中读取数据，然后封装成 DCData 返回。

为了使算法的具体实现与数据结构分离，本项目设计了多个算法类，其中 DCEngine 是实际应用中的顶层类，充当了枢纽的角色，它的 layout 方法接收 DCData 的指针 data 和参数结构的指针 args，返回自动布置的结果，用 DCPoint 的向量来表示。在 DCEngine 的 layout 方法中，需要调用一些重要的算法子模块，这些子模块也用类的方式各自封装起来，其中包括 DCAClassLocator 和 DCBClassLocator，分别负责自动布置单个楼层的 A 类引下线位点和 B 类引下线位点。

为了能够生成可视化结果，DCIO 还提供了 write 方法，该方法负责将 DCEngine 的处理结果转换成 geojson 字符串并写入文件当中，以便测试端进行分析和调试。

类图中只列出了主干方法，一些辅助性的方法没有列出。

4.潜在问题

暂无。

5.开发配置

编程语言：C++

编程环境：Visul Studio 2013

所需软件包：boost_1_70_0-msvc-12.0-64、CGAL 4.9.1 64 位

撰写人：郑友怡团队 殷俊麟

撰写时间：2021 年 5 月 31 日星期一