任务六：郊区环境下点云地面分割

说明文档

整体描述

机载 LiDAR 可以获取快速、低成本地获取大区域的高精度地形测量值。为了获取高精度 DTM/DEM 需要区分测量点中的地面点（由地面直接返回）及非地面点（建筑、车、植被）。众多学者采用了各种各样的方法来进行"点云地面点滤波"。本方案基于 Progressive Morphological Filter 算法。

2. Morphological Filters（形态学滤波）

2.1 膨胀/腐蚀

膨胀/腐蚀是其中的两个基础操作，通俗的说膨胀/腐蚀可以扩大/减小特征的尺寸，并以此组合为开/闭操作。针对 LiDAR 测量点 p(x，y，z)，高程 z 值在(x，y)处对应的膨胀操作可以定义为：

$$d_p = \max_{(x_p, y_p) \in w} (z_p)$$

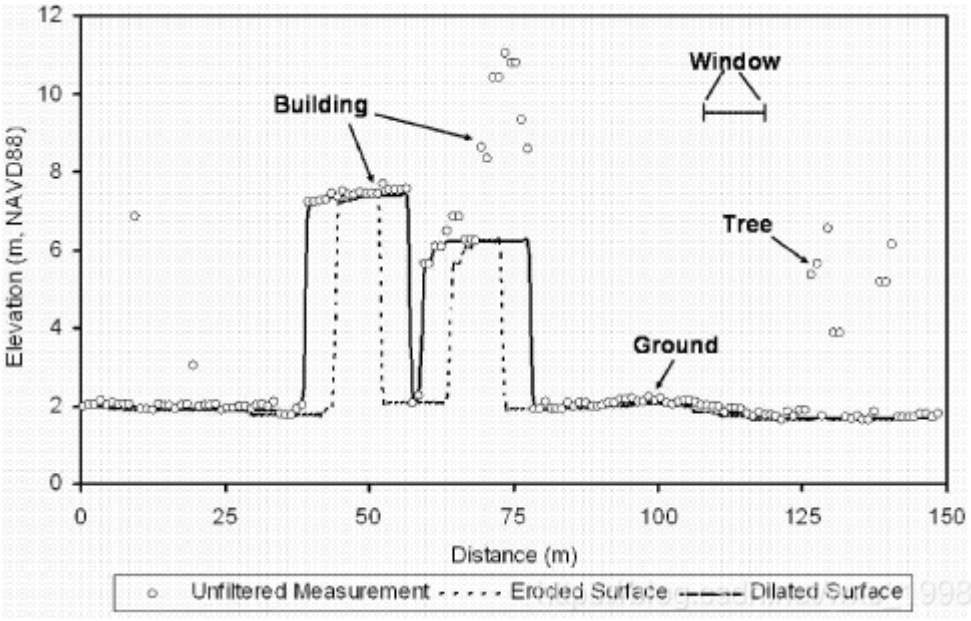
式中：(xp，yp，zp) 代表 p 点的相邻点，w 为操作窗口(可以为一维“线”也可以为二维“矩形/圆/其他形状等”)。膨胀操作完成后会输出 p 点在窗口 w 内具有最大高程值的近邻点。

与之类似的，腐蚀操作为在 p 点窗口 w 内找到具有最低高程值的近邻点。可以通过下式进行定义：

$$e_p = \min_{(x_p, y_p) \in w} (z_p).$$

任务六：郊区环境下点云地面分割

了解膨胀/腐蚀这两个基础操作之后，可以通过对其进行简单组合来来形成开/闭操作，其中开操作为先进行腐蚀再进行膨胀操作，而闭操作为先膨胀再进行腐蚀。在 LiDAR 数据处理中使用了“开”操作，处理效果如下图中所示：



可以从图中得知：“虚线”是先进行“腐蚀”操作所形成的表面，这个表面剔除了“树木”点，但是大型建筑物却变得不完整。“实线”是对“腐蚀”操作结果进行“膨胀”操作所形成的表面，可以看出其又恢复了大型建筑的形状。基于此，我们可以知道，“开操作”具备去除地面上的细小地物，保留大型地物的能力，这种能力对于后续处理是非常重要的。

2.2 形态学滤波

上述的“开操作”只是去除了细小地物，保留了大型地物，并没有去除所有非地面点去除，而且仅仅通过一个“开操作”也不可能实现对复杂地表的提取。因此，怎么利用好“开操作”的能力进行下一步骤的提取是进一步提升的关键。

Kilian 等人提出，可以在“开操作”处理后的结果中的每一个“窗口”内找到一个“最低点”，然后此窗口内的其他点若落在“最低点”的一个高程范围内则

认为是地面点。这个高程范围通常根据机载 LiDAR 系统扫描的精度来定义，正常为 20-30cm。

此方法中有两个方面对最后的结果好坏非常重要：

1.滤波窗口的尺寸，如果窗口尺寸设置的比较小，则可以很好的保留地面起伏的细节，但是只能去除像汽车、树木等细小地物，而对建筑物则去除效果较差（屋顶通常被认为是地面）。相反，若窗口尺寸设置的较大，则会过度去除一些“地面点”，例如，一条道路与一条小水沟相邻，若窗口尺寸大于道路的宽度，则道路可能就会被认为是非地面点（因为小水沟中的点高程较低，会被认为是窗口内的最低点，而道路点较高，被判断为非地面点）。此外，一些局部的小山丘、沙丘都极可能被“切除”。

2.建筑与树木在特定/局部区域的分布。

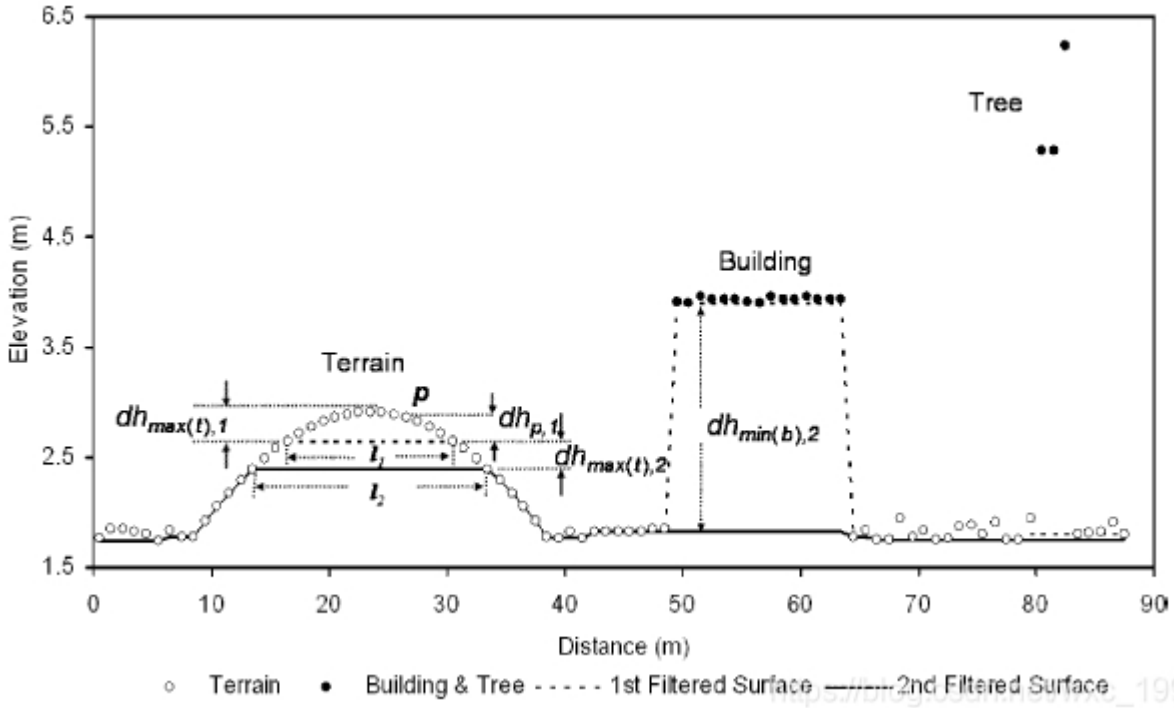
注：一个最理想的情况是我们可以设置一个“窗口”，这个“窗口”的尺寸可以足够小，能够保留地面细节。同时，还可以足够大，能够去除建筑、汽车、树木等地物。但是，这种理想情况在实际数据集中国并不存在。

为了解决这一问题，Kilian 等人继续提出了可以通过改变窗口大小来多次进行滤波。每个点都被赋予一个与窗口大小相关的权重，窗口尺寸越大，点的权重越高。这种方法虽然得到了更好一些的效果，但是没有从"point level"进行区分地面点与非地面点。（"point level"区分的地面点与非地面点之后可以通过插值的方法使得 DEM/DTM 的生成效果更好。）

3. Progressive Morphological Filters

任务六：郊区环境下点云地面分割

由上述 2.2 节中的分析可知 ,传统的形态学滤波很难通过一个固定大小的窗口去检测出各种尺寸变化的不同地物。这一问题可以通过逐渐改变窗口大小来解决。如下图中所示，首先使用一个尺寸为 11 的窗口来对原始数据进行开操作。由图中的“虚线”可以看出树木等尺寸小于 11 的地物被去除，且地形特征中小于 11 的部分被“切除”（山丘顶部高程被替换成了 11 中的最小值），但是，尺寸大于 11 的建筑物被保留了下来。接着，进行下一次迭代，窗口尺寸变为 12，对上一次的处理结果进行开操作处理，结果从“实线”中可以看出，12 大于建筑的尺寸，所以建筑也被去除，但同时山丘顶部被“切除”的范围更大。



需要注意的是： 通过逐渐增加窗口尺寸解决了去除不同大小地物的问题，但是又引入了"山丘"顶部等小于窗口尺寸的地形特征部分被“切除”的问题。

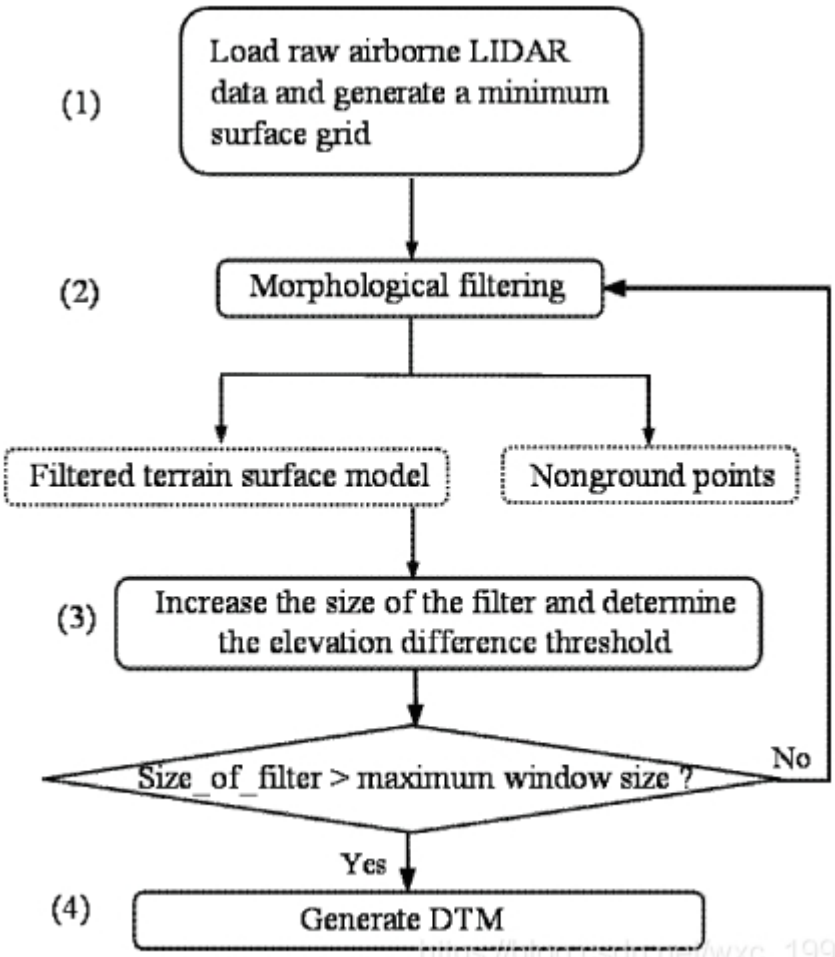
为了解决这一新出现的问题，可以通过引入一个高度差阈值来解决。建筑屋顶和地面点之间的高程差是“突变”（abrupt change），而地面高程是“渐变”（gradual change）。因此，二者之间高程变化中的明显区别可以帮助我们进行区分。假设 $dh_{p,1}$ 代表原始 LiDAR 测量值与在任意给定 p 点处第一次迭代表面之间的高程差，$dh_{T,1}$ 代表高程差阈值，则如果 $dh_{p,1} \leqslant dh_{T,1}$ 点 p 就被认为是地面点，反之如果 $dh_{p,1} > dh_{T,1}$ 就认为点 p 是一个非地面点。此后，令 $dh_{max(t),1}$ 为当前迭代中初始地面点与滤波表面之间差值的最大值，则如果选取的 $dh_{T,1} > dh_{max(t),1}$ 则所有的测量值都会保留。

在第二次迭代中假设上一次滤波表面和本次滤波表面的最大高程差为 $dh_{max(t),2}$，则如果 $dh_{max(t),2} < dh_{T,2}$，则高程差值在 $dh_{max(t),2}$ 范围内的地面点都会被保留。类似的，假设在上次迭代和本次迭代之间建筑高程差值最小为 $dh_{min(b),2}$（通常近似为建筑的高度），如果 $dh_{min(b),2} > dh_{T,2}$，则建筑就会被移除。

通常设置 $dh_{T,k}$ 为研究区域第 k 次迭代中建筑物的最矮高程值。以 $dh_{T,k}$ 作为阈值，对于第 k 次迭代中的任意点 p 如果 $dh_{p,k} < dh_{T,k}$ 则将其标记为地面点，反之为非地面点。通过这种方式，不同尺寸的建筑物（树）可以随着迭代窗口尺寸的增加逐步被去除。

综上所述，Progressive Morphological Filters 的详细流程如下图所示：

任务六：郊区环境下点云地面分割



我们可以对上图总结以下四个步骤：

加载不规则点云，划分为规则网格，在每个网格中选取高程最低点（如果网格中没有点则根据最近邻点进行插值），构建最小高程表面。

使用输入的初始滤波窗口尺寸、1）中获取的最小高程表面作为第一次迭代的参数进行第一次迭代。随后，在后续的迭代中，以前一次获取的滤波表面及 3）中计算的滤波窗口尺寸作为输入。每次迭代的输出有两部分：a）形态学滤波后得到的更加光滑的表面；b）基于不同阈值所检测出来的非地面点。

计算新的滤波窗口尺寸及不同的高程插值阈值。重复步骤 2)、步骤 3)直到

窗口尺寸大于预设的最大窗口。

基于去除非地面测量值的数据进行生成 DEM/DTM。

注意：每一次迭代中的“开操作”实际都是作用在步骤 1）所划分网格中的点，所以 Progressive Morphological Filters 是"point level"来对 LiDAR 测量值进行滤波处理的。

3.1 参数计算(窗口尺寸/高程差阈值)

在上述步骤 3）中我们要变化窗口尺寸 wk 和高程差阈值 dhT,k 两个参数的值，以进行下一次迭代，那么这两个值是怎么计算的呢？

3.1.1 窗口尺寸

首先是窗口尺寸 wk 有两种计算方式，第一种是：

$$w_k = 2kb + 1$$

式中，k 为迭代次数，b 是初始窗口大小（由用户进行输入），最后+1 是为了保证 wk 为一个奇数，窗口对称。但是，如果一个研究区域具有非常大的地物，这种增加窗口尺寸速度太慢则会耗费较多时间。因此，可以使用第二种方式，通过指数增长来改变窗口大小，计算如下式：

$$w_k = 2b^k + 1$$

同样的，式中 k 为迭代次数，b 是初始窗口大小（由用户进行输入），这种方式的增长速度较第一种方式快很多。

3.1.2 高程差阈值

高程差阈值与研究区域的地形坡度密不可分，因此可以通过下式进行计算：

$$dh_{T,k}=\begin{cases} dh_0, & \text{if } w_k \le 3 \\ s(w_k-w_{k-1})c+dh_0, & \text{if } w_k > 3 \\ dh_{\max}, & \text{if } dh_{T,k} > dh_{\max} \end{cases}$$

式中，dh0 为初始高程差阈值，s 为坡度，c 为格网大小，dhmax 为最大高程差阈值。

在城市区域，树木、汽车相对于建筑的尺寸小很多，所以通常是最后滤除建筑，最大高程差阈值 dhmax 可以设置为一个固定值（如最矮建筑物高度）。而在山区，主要的非地面点为植被，所以并没有必要设置固定的最大高程差阈值 dhmax，于是 dhmax 通常被设置为测区内的最大高程差。

此外，坡度 s 通过第 k 次迭代的最大高程差 dhmax(t),k，以及窗口尺寸 wk 进行计算，如下式所示：

$$s=\frac{dh_{\max(t),k}}{\frac{(w_k-w_{k-1})}{2}}.$$

解决思路

1. 本文基于 PCL 库 OpenMP 版本 Progressive Morphological Filter 代码修改。实际采集的激光雷达数据会有 nan 点，而在多线程版本中，由于 nan 点的存在会

导致计算出错；因此在遍历点云的时候我们加入了 nan 点的判断。先排除 nan 点再计算。

2. PCL 库中不存在点云数据结构类型 XYZIL，因此我们自定义了此数据结构。

架构设计

由于单线程版本运行很慢，而且大部分时间都是在读取 PCD 点云数据，在我们笔记本（CPU，I7-8750H 固态硬盘）上单线程实测读取 600ms 左右，地面滤波 50ms 左右。所以本文采用线程池管理，批量处理点云数据。伪代码如下：

读取配置文件

读取所有 PCD 文件名

将所有任务提交到线程池执行

对于每一个任务：

读取 PCD 点云数据

OpenMP 优化的 Progressive Morphological Filters 处理得到地面点索引

所有点云转换成数据类型 XYZIL

地面点对应的点 label 为 1

运行指令说明

与代码文件中的 Readme.md 文件内容一致

```
# 将测试文件放在宿主机器 /task6 文件下

sudo mkdir -p /task6

sudo mkdir -p /task6_result/spirt

sudo rm /task6_result/spirt/*

<!-- sudo cp *.pcd /task6 --|

# build docker 构建镜像
# 构建镜像 时将本文件内的大文件移出此文件夹，因为 dockerfile 会将本文件夹拷贝到容器中

sudo docker build -t p1:v2 .
```

任务六：郊区环境下点云地面分割

```
# 或者导入镜像
sudo docker load -i pl_v2.tar
# run 运行容器
sudo docker run --rm -it -v /task6:/task6 -v /task6_result/spirt:/task6_result/spirt

pl:v2 /usr/bin/bash /root/plane/run.sh
```

相关指标

对于提供的 39 个 PCD 文件，我们处理时间约 10 秒（点云保存形式为 ascii

）

待改进的方面：

1. Progressive Morphological Filter 算法中直接将 lable 赋值

2. 读取 pcd 采用 XYZIL 类型而不是 XYZI 类型。

参考资料

点云地面点滤波(Progressive Morphological Filter)算法介绍（PCL 库）