单位代码 10006
学　　号 15091053
分 类 号 O244

北京航空航天大學
BEIHANG UNIVERSITY

毕业设计 (论文)

基于机载 LiDAR 数据的城市房屋结构提取与重建方法研究

学 院 名 称　数学与系统科学学院
专 业 名 称　统计学
学 生 姓 名　杨启航
指 导 教 师　姜鑫

2019 年 06 月

北京航空航天大学

本科生毕业设计（论文）任务书

I、毕业设计（论文）题目：

基于机载 LiDAR 数据的城市房屋结构提取与重建方法研究

II、毕业设计（论文）使用的原始资料（数据）及设计技术要求：

一份城市点云 LiDAR 数据

基于 PCL 的 C++ 编程

III、毕业设计（论文）工作内容：

点云数据处理

建筑点云的提取与分离

建筑结构提取

门窗结构提取

城市点云模型重构

IV、主要参考资料：

魏征. 车载 LiDAR 点云中建筑物的自动识别与立面几何重建

牛路标. 基于机载和车载 LiDAR 数据的建筑物三维建模方法研究

王庆栋. 基于语义建模框架的机载 LiDAR 点云建筑物三维重建技术研究

S. Tuttas U. S. Reconstruction of Rectangular Windows in Multi-looking Oblique View ALS Data

数学与系统科学 学院 统计学 专业类 150923 班

学生 杨启航

毕业设计 (论文) 时间： 2019 年 03 月 04 日至 2019 年 06 月 03 日

答辩时间： 2019 年 06 月 03 日

成　　绩：

指导教师：

兼职教师或答疑教师（并指出所负责部分）：

系（教研室）主任（签字）：

注：任务书应该附在已完成的毕业设计（论文）的首页。

本人声明

我声明，本论文及其研究工作是由本人在导师指导下独立完成的，在完成论文时所利用的一切资料均已在参考文献中列出。

作者： 杨启航

签字：

时间： 2019 年 06 月

基于机载 LiDAR 数据的城市房屋结构提取与重建方法研究

学　　生：杨启航
指导教师：姜鑫

摘　要

随着智能小区、数字城市等新概念的产生，人们对于更加逼真的城市 3D 模型重构提出了更高的期望。然而，城市点云模型重建研究，大多重点在于如何将建筑构建的更加准确，而忽略了对于建筑上更细节结构，比如门窗的重构。本文基于机载 LiDAR 数据，详述了城市点云模型重构的步骤与方法，并在此基础上，研究对于房屋结构，主要是门窗结构的提取与重构的方法。本文使用点云库（Point Cloud Library，PCL），将所述算法实现。我们基于一个城市点云数据，对点云进行了包括去噪、稀疏化和参数化的预处理，然后提取并剔除了地面点与非建筑物点，将建筑物聚类分离，对每个建筑提取立面，将屋顶用于构建建筑模型，将墙面用于提取门窗。本文将各步骤的成果与最终成果展现出来，并且分析各方法的实现效果，对城市点云模型重建中遇到的问题提出解决方法。

关键词：机载 LiDAR，城市模型重建，房屋结构，门窗提取

Research on extraction and reconstruction of the structure of urban building based on airborne LiDAR data

Author: Qihang Yang
Tutor: Xin Jiang

Abstract

With the wild applications of data-driven concepts such as smart communities and digital cities, people have raised significant expectations for more realistic urban 3D model reconstruction. Currently, most of the researches on urban point cloud model reconstruction mainly deal with the accuracy of the reconstruction model, while the detailed structure of the building, such as the reconstruction of doors and windows, is usually ignored. Based on a given airborne LiDAR data set, this paper demonstrates the steps and methods of urban point cloud model reconstruction, and further, studies the method of housing structure reconstruction, mainly about the extraction and reconstruction of doors and windows structure. Concretely, we introduce several common used algorithms from Point Cloud Library (PCL). Based on the city point cloud data, we perform procedures including denoising, sparseness and parameterization on the point cloud, then extract and eliminate ground points and non-building points. Based on this, we could cluster the buildings separately for each building，extracting the façade and using the roof to rebuild the building framework. Finally, we use the wall to extract the doors and windows. In this paper, we present the results of each step, and compare the effects of various methods, we also propose solutions to the problems encountered during the reconstruction of urban point cloud models.

Key words: LiDAR, urban reconstruction, doors and windows

目　　录

1 绪论

1.1 研究背景及意义

随着信息技术与通讯技术的发展，近年来，人们生活逐渐向着更加智能化的方向发展。当智能与自动化与生活中的方方面面相结合，诞生了例如自动汽车、智能小区、数字城市等多方面的科技产品与科技蓝图。作为智慧城市的基础，准确的城市数字化 3D 模型有着非常大的重要性。详细且精准的城市模型，对于自动汽车的自动行驶、智慧城市的数据管理等都有着至关重要的作用。

传统的城市模型构建，只是单纯的取决于道路与房屋的地理坐标。基于城市规划信息，确定城市各道路、房屋和其他非建筑物的位置，然后构建二维地图模型。再进一步的，根据各房屋的规划设计信息，单独构建每栋建筑模型，然后再拼接成城市模型。这样的方法，不仅耗时耗力，而且在构建结果上十分的不准确，只能保留很少的一部分城市信息。这对于更精细化的智慧城市建设肯定是不利的。

随后，人们开始将激光雷达技术（LiDAR，Light Detection and Ranging）应用与城市模型的构建中。激光雷达技术，即利用雷达接收由发射器发射的无线电波，以此记录反射物体坐标的技术。常用的有机载 LiDAR 和车载 LiDAR，即用无人机从高空扫描城市，和用汽车搭载设备在城市中行驶扫描城市。机载 LiDAR，因为其扫描速度快，扫描范围广，并且受非建筑物遮挡较少，较多地运用在大范围的城市模型重建工程中。

图 1.1 机载 LiDAR

综上所述，利用机载 LiDAR 采取到的点云数据，构建城市模型，对于自动汽车、智慧城市等更科技蓝图的更好实现，有着非常重要的意义与研究价值。而如何能够使重建更加具体准确，也就成为了相关研究的重点与难点。

1.2 国内外研究现状

近十年来，国内外都有不少高校与学者从事城市点云模型重建的研究中，人们研究的重点，主要由如何更加智能与自动地构建城市模型，发展到如何更加细致准确地构建模型。张栋（2005）[1] 将 LiDAR 点云数据和通过无人机采集的航空影像中的房屋轮廓信息相结合，相辅相成构建城市模型。喻亮（2011）[2] 提出了基于多维空间相似度的点云分割方法和基于地物特征模糊度量和可信度判别的分类识别方法，以及弧面形态的地物对象特征提取方法，对特定特征房屋特点提取更加准确。魏征（2012）[3] 基于 RANSAC 平面分割提出了对建筑物点云立面几何位置边界和建筑物立面细节结构的自动提取的方法。牛路标（2016）[4] 改进了 RANSAC 算法实现了屋顶点云的分割和建筑物屋顶模型的重建。张昌赛等（2017）[5] 提出了利用布料模拟滤波（CSF）算法对复杂地形进行滤波。王庆栋（2017）[6] 提出了一套基于 XML 的建筑体描述语言的面向建筑三维重建的语义建模框架，将城市建模语义化。

国外也有很多学者从事城市点云模型重建的相关工作，并且取得了不错的进展。G. Sithole 等（2003）[7] 提出了对于近地面物体，如桥梁、斜坡等特征的提取与重建。V. Verma 等（2006）[8] 提出了基于特征值对非建筑点云的剔除和基于拓扑结构对房屋结构的分块构建。A. Golovinskiy 等（2009）[9] 根据各个物体特征，精确分离出包括树木、车辆、信号牌、海报、邮筒等多种非建筑物物体。S. Pu 等（2009）[10] 提出了一种基于边界点将建筑墙面模型自动提取出来的方法。S. Tuttas 等（2012）[11] 提出了利用分割楼层和基于象限的方法提取门窗结构。

纵观城市点云模型重建的研究历程，我们可以看出，人们的研究由浅入深，由宏观到具体，从原本研究如何提取建筑、非建筑，到后来提取立面、屋顶。渐渐地，研究中心现如今放到了如何提取立面中的细节，例如门窗等特征物体上。这时目前城市点云模型重建研究中相对成果较缺失的部分。

本文也就着重于这一缺失点，在基本完成城市点云模型重建的一般过程之后，深入探讨对于建筑物门窗的提取与重建方法。

1.3 研究内容和方法

查阅城市点云模型重建的相关研究我们可以发现，对于城市点云模型重建一般有着一套较为成熟的流程。这个流程可以大体总结如下：首先，获取点云，一般会采用机载 LiDAR 或者车载 LiDAR 来采集点云数据；收集到的点云会进行一定的预处理，例如去噪等相关工作，以方便之后的操作；处理过后的点云，会先将地点剔除，这有助于将地

面上的所有物体大致上分离开来；之后，通过相关操作将非建筑物剔除，比如树木等不规则物体；在得到了几乎只剩各个建筑物的物体之后，一般的流程是先提取屋顶的点云；之后通过一定的方法提取出屋顶的轮廓，找出轮廓顶点；最后根据顶点与地面，构建建筑物体状模型。

这样构建的模型十分的简单，清楚，基本能够把各个建筑物位置、大小等相关信息包括其中。但是也相对粗燥，建筑模型过于简单，缺少屋顶、门窗、墙壁纹路等细节。近些年来城市点云模型重建的研究也就大多集中在如何表现更多细节上，而本篇论文也就着重在门窗的提取与重建上。

因为各个建筑物、建筑物各个立面的门窗信息都不一定相同，所以我在一般操作流程上进行了一定的改进，便于我的工作的展开。在提取屋顶之间，先将建筑物逐个聚类，分离开来，这样便于将各个立面分割开来。屋顶平面继续进行建筑大体模型的重建，其余墙面点云则用于门窗提取的相关工作。

下图就是一般城市点云模型重建的流程图和我的流程图。

图 1.2 一般城市点云模型重建流程

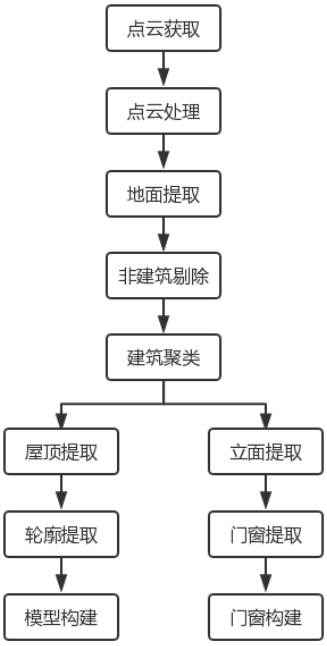


图 1.3 我的城市点云模型重建流程

本文对于点云数据的研究与操作，是基于点云库（Point Clouds Library, PCL）的。PCL 是一个大型跨平台的开源 C++ 编程库，它实现了大量点云相关的算法，涉及获取、滤波、分割、匹配等等多种操作。本文基于 C++ 语言，使用 PCL 库，对目标点云进行

操作与研究，各步骤结果与最终结果都将在本文中展出。

1.4 论文的组织结构

本文各章主要内容概括如下：

第一章为绪论，主要是阐述本文研究的背景，包括问题的提出、研究的意义、国内外各学者的研究现状，以及介绍本研究将会用到的重要工具。这一部分详细介绍了城市点云模型重建这个研究方向的大体情况，对该方向研究给与读者一个基本但充分的了解。

第二章为数据预处理，对于获取到的城市点云数据，一般需要进行一定的操作之后才能够方便进行操作，这一部分涉及去噪、稀疏化、参数化等操作。我们将解释各个处理的原理，将算法实现并且分析效果。

第三章为面片提取，正如上文我的流程所示，我们将对处理过后的目标点云做立面提取。将提取到的立面一一保存，以方便接下来的一步步操作。这一部分将介绍立面提取的相关算法，并且以某一栋建筑为例，进行实现。

第四章为门窗提取，本文的创新点基本也着重于这一部分，即在城市模型重构的基础上，提取建筑上的门窗信息，以使建筑模型更加丰富。这一部分讲解了门窗提取的方法，并且分析其中可能遇到的困难与处理方法。

第五章为模型重建，我们将选用一个方法，将建筑模型构建出来，并且将前文得到的门窗信息也体现出来。这一部分将介绍建筑重构的常用方法，并且将重建后的模型展现出来。

之后是结论章，我们将对全文做一个总结，整理全文思路，并且对本文的成果做一定的分析。这一部分将会总结本文的亮点，也会提出本文需要改进的地方。

最后就是我的致谢与本文的参考文献和附录。

以上就是本论文的结构与各章内容，下面我们将正式进入城市点云模型重构的研究中。

2 数据预处理

2.1 点云数据获取

在第一章中，我们介绍了机载 LiDAR 和车载 LiDAR 的区别。本文选用机载 LiDAR 扫描的一片城市数据作为研究对象，下图就是该片点云的图片。该片点云数据较为完整，建筑物大体轮廓清楚，各建筑物之间空隙较大，方便聚类操作。部分立面上门窗空洞明显，便于提取操作的实施，同时也有部分立面上有较大遮挡空缺，这也符合一般的机载 LiDAR 扫描数据的特性，这部分点云便于做空缺填补的相关研究。总之，这是一个非常利于研究，也非常有研究价值的点云数据。

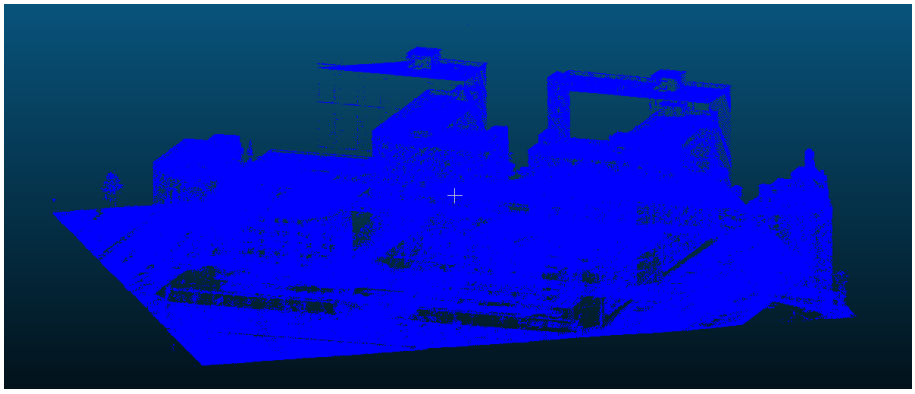


图 2.1 机载 LiDAR 扫描的点云数据

在得到点云数据之后，会对点云数据做一个预处理工作，使数据处于一种方面操作的状态。点云数据预处理是有一定的常用方法的，一般可以概括为三步：去噪、稀疏化和参数化。

2.2 点云去噪

去噪，即是去除噪音点。通过一定的方法，将不在目标物体点云内的点去除。如何定义和挑选出噪音点有很多种方法，例如统计滤波算法、低通滤波算法、移动最小二乘 (MLS) 曲面拟合算法和基于偏微分方程 (PDE) 的曲面逼近算法等。

近些年来，有许多学者在如何更好地去噪上取得了一定的成果。M. Alexa 等（2003）[12] 利用最小二乘曲面拟合点云模型从而实现对噪声点的剔除。C. Lange 等（2005）[13] 提出了基于偏微分方程 (PDE) 的曲面逼近算法。刘大峰等（2007）[14] 利用鲁棒性的统

计方法，根据核密度将噪声去除。

在这里，为了追求操作的速度和便捷，我们尝试两种较为简单直接的去噪方法：直通滤波去噪法和统计滤波去噪法。

2.2.1 直通滤波去噪

LiDAR 的原理，是通过激光，即原子受辐射释放的能量以光子的形式放出，来定位对应物体的位置。这种定位方式，很容易受到空中漂浮的颗粒和水滴等物体的影响。所以一般的机载 LiDAR 扫描数据，会有很多漂浮在空中的多余噪音点，在下图左可以看出。所以第一步，我们将在垂直高度上过高和过低的点去除。

已知每个点都有一个坐标，设为 $P(x, y, z)$。我们将所有 $z < \theta_1$ 和 $z > \theta_2$ 的点都去除。其中 θ_1 和 θ_2 由地面和屋顶点云决定。结果如下图右所示。

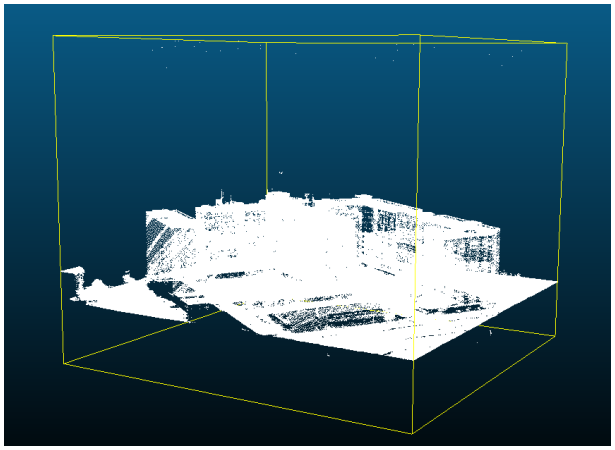


图 2.2 直通滤波去噪前的点云

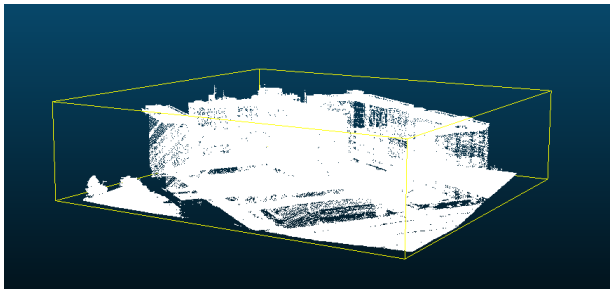


图 2.3 直通滤波去噪后的点云

可以看出空中的点和地面以下的多余的点都已经被删除了。

2.2.2 统计滤波去噪

另一种非常常用的去噪方法，统计滤波去噪法，即通过统计周围点云距离的方式，按照一定的统计分布，定义离群点，以达到去除噪音的作用。对点云中的每个点 P，对该点建立所有点云的 k-d 树，取周围一定范围内的点作为领域 Σ，对 Σ 中的任意点 Q_i，计算 Q_i 到 P 的距离 d

$$d(P, Q_i) = \sqrt{(x_P - x_{Q_i})^2 + (y_P - y_{Q_i})^2 + (z_P - z_{Q_i})^2}. \tag{2.1}$$

我们假设 Σ 中的任意点到 P 的距离是服从正态分布 $N(\mu, \sigma^2)$。则可以估计

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} P(P, Q_i). \tag{2.2}$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} (d(P, Q_i) - \hat{\mu})^2. \tag{2.3}$$

其中，设 $\Sigma = \{Q_1, Q_2, \cdots Q_n\}$。这时，$\Sigma$ 中所有到点 P 距离在 $(\hat{\mu} \pm k\hat{\sigma})$ 之外的点，就可以被定义为是离群点。常数 k 视情况而定。

根据正态分布标准偏差的性质，我们会尽可能多的保留了原点云中的数据特征，又把明显的离群点剔除，以达到降噪的效果。

然而在我们用于研究的点云中，统计滤波去噪的效果并不是特别好。因为扫描机器本身就较为精准，并没有太多的离群点。相反的，强行使用统计去噪反而把较为稀疏部位的立面和门窗点云给去除掉了。这是不希望看到的，所以我们在实际操作中略去了统计滤波去噪这一步。

2.3 点云稀疏化

在城市点云的相关研究中，点云数量一般都会非常巨大。点云密度很大，并且分布不均匀，这不仅对后续我们进行曲面构建增加了工作量，而且还可能出现曲面重叠等错误情况。所以，一般情况下我们不必保留原有的点云密度，而可以选择对点云做一个稀疏化，以减少程序运行压力。

常见的点云稀疏化方法包括最小距离法、均匀采样法、曲率精简法和均匀与非均匀网格法 [15]。这里我们选用均匀网格法。

均匀网格法，又称为体素化网格法。指对点云数据创建一个三维体素栅格，然后在每个体素内，用的重心点来替代所有点。这样就可以实现对整体点云数据的稀疏化。不过体素栅格如果过大会导致稀疏后点云与原点云产生一点小的偏差，体素栅格过小稀疏意义又不太明显，故体素栅格的大小需要视实际情况而选择。

设对点云构建边长为 l 的正方形栅格。则对任意点 $P(x, y, z)$，P 对应的栅格坐标为

$$(a, b, c) = \left(\left[\frac{x - x_{min}}{l}\right], \left[\frac{y - y_{min}}{l}\right], \left[\frac{z - z_{min}}{l}\right]\right). \tag{2.4}$$

其中 x_{min}、y_{min}、z_{min} 代表点云所有点的 x 轴、y 轴、z 轴坐标的最小值，$[\cdot]$ 代表取整。将所有点与栅格对应起来之后，对任意栅格 (a_t, b_t, z_t)，取该栅格中的所有点，用点 $P_t(x_t, y_t, z_t)$

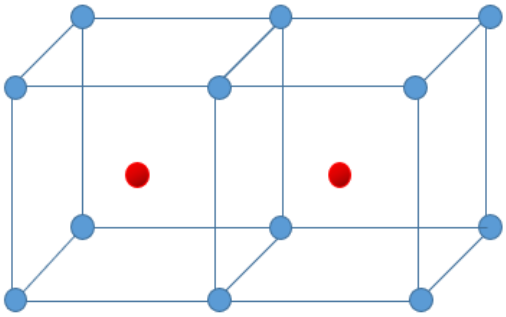


图 2.4 体素化网格法示意图

来代替整个栅格。点 P_t 的坐标由下式给出

$$x_t, y_t, z_t = \left(\frac{1}{n} \sum_{i=1}^{n} x_i, \frac{1}{n} \sum_{i=1}^{n} y_i, \frac{1}{n} \sum_{i=1}^{n} z_i \right). \tag{2.5}$$

其中 $(x_i, y_i, z_i), i = 1, 2, \cdots, n$ 是栅格 (a_t, b_t, z_t) 中所有点的坐标。

下图是目标点云数据稀疏化之后的结果。

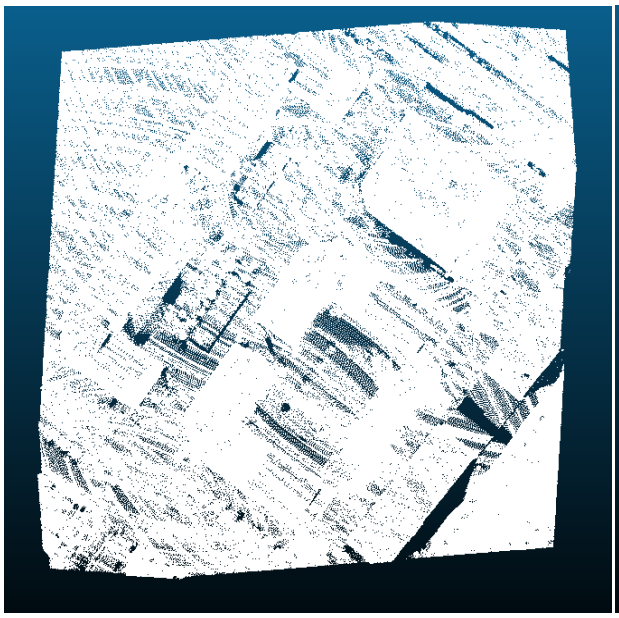


图 2.5 稀疏化前的点云

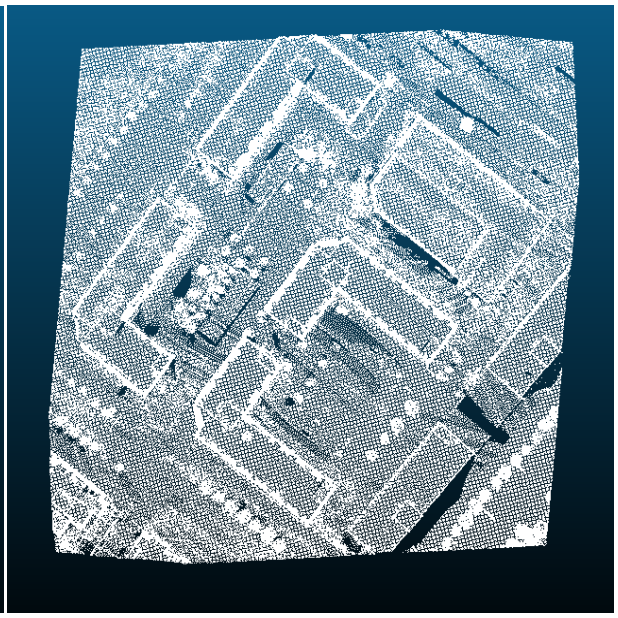


图 2.6 稀疏化后的点云

稀疏化之前，原点云有 1863672 个点，稀疏化之后，点云还剩 376114 个点，压缩到了只剩 20% 的点。可以很明显的看出，稀疏化后的点云分布更加均匀，不再具有原点云中的明显的机载 LiDAR 扫描纹路。较少的点云数量也对之后的程序操作减轻了很大的压力。

2.4 点云参数化

经过点云稀疏化之后，点云的数量较有减少，但是在一般情况下，点云本身采集时的不准确以及稀疏化时产生的偏差，导致点云“层次起伏”。为了便于之后的模型构建，经常会采用取局部点云，用参数曲面拟合，然后在参数曲面上重取样，在误差尽可能小的前提下使得点云更加的平滑化。

对于空间中散乱点云数据的曲面拟合，我们常见的方法有基于最小二乘法、正交最小二乘法和移动最小二乘法等的曲线拟合方法 [16]。这里我们选用移动最小二乘（MLS）法，该方法具有较好的平滑性和拟合质量 [17]。

假设在一个拟合区域的局部子域上，拟合函数 $f(x)$ 表示为

$$f(x)=\sum_{i=1}^{m}\alpha_i(x)p_i(x)=p^T(x)\alpha(x). \tag{2.6}$$

式中 $\alpha(x)=[a_1(x),a_2(x),\cdots,a_m(x)]^T$ 为待求系数，$p(x)=[p_1(x),p_2(x),\cdots,p_m(x)]^T$ 称为基函数。考虑加权离散 L_2 范式

$$\begin{aligned} J&=\sum_{i=1}^{n}w(x-x_i)[f(x)-y_i]^2\\ &=\sum_{i=1}^{n}w(x-x_i)[P^T(x_i)\alpha(x)-y_i]^2. \end{aligned} \tag{2.7}$$

式中，$w(x-x_i)$ 是节点 x_i 的权函数，对式（2.7）取极小值

$$\frac{\partial J}{\partial\alpha}=A(x)\alpha(x)-B(x)y=0. \tag{2.8}$$

$$\alpha(x)=A^{-1}(x)B(x)y. \tag{2.9}$$

其中

$$A(x)=\sum_{i=1}^{n}w(x-x_i)p(x_i)p^T(x_i). \tag{2.10}$$

$$B(x)=[w(x-x_1)p(x_1),w(x-x_2)p(x_2),\cdots,w(x_n)p(x_n)]. \tag{2.11}$$

将式（2.9）带入式（2.6），就可以得到 MLS 拟合函数，表示为

$$f(x)=\sum_{i=1}^{n}\phi_i^k(x)y_i. \tag{2.12}$$

下图就是参数化前的点云与参数化后的点云的对比图。

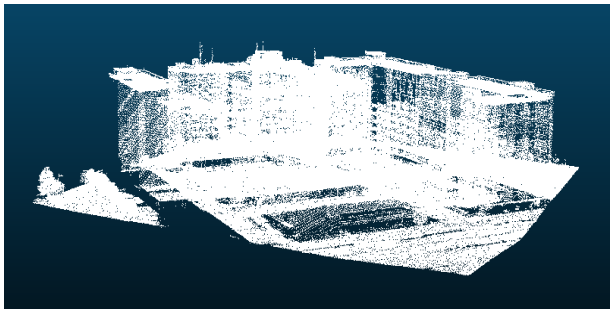


图 2.7 参数化前的点云

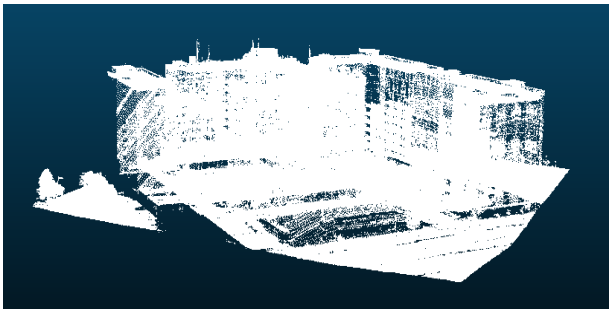


图 2.8 参数化后的点云

图片从整体上看并不能看出太大的区别，但是在细节上还是有不少改变的，主要体现在平面点云调整在一个平面内，这对于之后提取平面等操作提供了很大的帮助。

至此，对点云数据的收集与预处理就基本全部完成了，我们已经得到了一份较为合适用于研究的城市点云数据。接下来将开始对点云数据做各种分析与操作，以提取出城市建筑的各项数据信息，来进行模型的重建。

3 面片提取

3.1 地面点提取

在完成了点云的预处理之后，我们得到了方便操作的研究对象，就可以正式开始进行城市建模的相关工作了。一片城市点云一定是连在一起的，而城市建模的基础就是特征提取，在整体的点云数据上肯定是很难操作的，我们要尽可能的把点云中的各个部分分离开来，以方面每一个部分的特征提取。

首先第一步，一般都是将地面点云提取出来。显然，在一般情况下，能连接城市点云最多部分的一定是地面点云，它基本将建筑点云、非建筑点云几乎所有的物体点云都连接在了一起。如果能够把地面点云提取出来并且把它们去掉，那各个建筑与各个非建筑物之间很可能就脱离开来了，这是非常有帮助的。

多数情况下，地面都是整个点云数据中最大的一个平面，我们如果用平面去拟合整个点云数据，地面一定是拟合程度最好的一部分点云。所以，这里我们选用 RANSAC 算法，来提取地面点云。

3.1.1 RANSAC 算法

RANSAC 算法，全称 Random Sample Consensus 算法，中文名为随机抽样一致性算法。由 M. A. Fischler 和 R. C. Bolles 在 1981 年提出 [18]，是一种模型拟合算法。它通过随机取样的方式，确定最大、误差最小的符合某种模型的点云子集。该算法描述如下：

给定一个需要至少 n 个数据点来确定自有参数的模型，以及一组数据点 P，要求 P 中的点数大于 n，即 $\#P \geq n$

1. 随机选择一个来自 P 中的 n 个数据点的子集 S_1 并拟合该模型;

2. 根据误差容限，使用拟合的模型 M_1 来确定 P 中的子集 S_1^*，其中 S_1^* 称为 S_1 的共识集；

3. 设定误差函数 $g(\cdot)$，如果 $g(S_1^*)$ 大于误差阈值 t，则使用 S_1^* 计算新模型 M_1^*；如果 $g(S_1^*)$ 小大于误差阈值 t，则随机选择新的子集 S_2，重复上述步骤；

4. 如果在经过一些预定次数的试验后，没有误差小于 t 或有更多成员的共识集，则已有最大共识集为目标子集，或者视为失败终止。

通过 RANSAC 算法，我们可以提取出原点云中，最大、最符合平面拟合的点云子集。在地面点提取中，则自然就是地面点和其所在平面。在之后的操作中，该算法还被

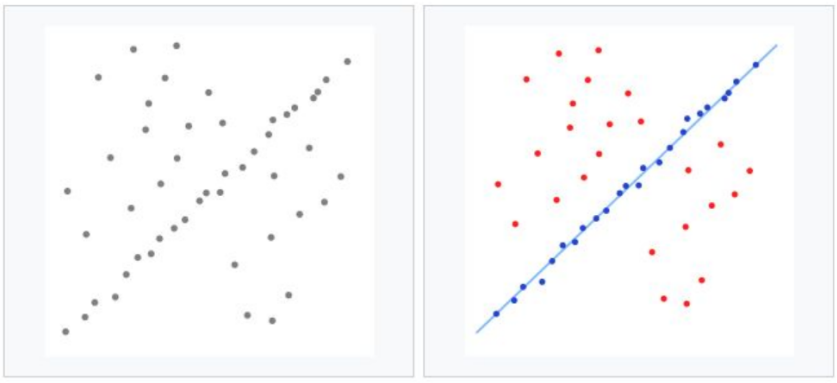


图 3.1 RANSAC 算法示意图

用于提取立面，这将会在届时提到。

3.1.2 地面点剔除

我们利用上文所说的 RANSAC 算法对目标点云做平面提取，提取出地面所在平面参数，并且把所在该平面的点都提取了出来。为了更大程度的把所有地面点纳入其中，并且考虑地面点云非单层对平面拟合产生的影响，我们把在拟合平面一定范围内的点都定义为地面点，以方便将地面上物体点云彻底分离开。

在 RANSAC 算法中，我们可以得到拟合平面的参数。设地面所在平面方程为

$$Ax + By + Cz + D = 0. \tag{3.1}$$

遍历目标点云中的所有点，对任意点 $P(x_0, y_0, z_0)$，当点到平面的距离

$$d = \frac{|Ax_0 + By_0 + Cz_0|}{\sqrt{A^2 + b^2 + C^2}}. \tag{3.2}$$

满足 $d \leqslant \theta$ 时，其中 θ 为固定阈值。那么我们将该点定义为地面点。

下图就是提取出的地面点和除去地面点之后的点云。可以看出每个建筑与非建筑物都基本分离开来了。这对我们之后的操作提供了很大的方便。

图 3.2 地面点

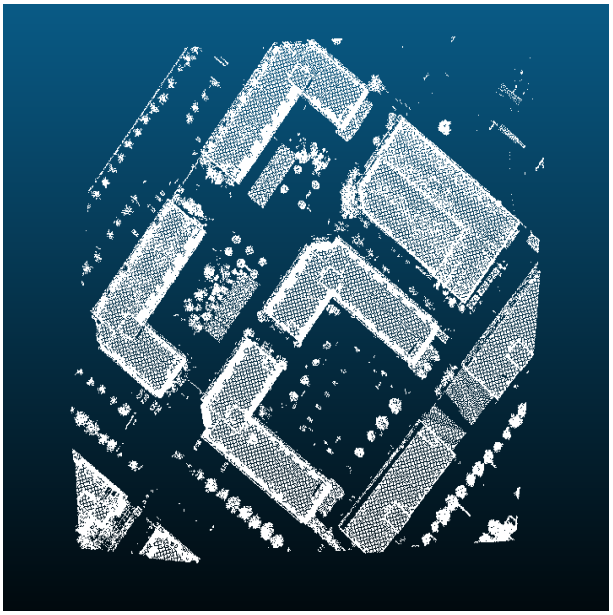


图 3.3 非地面点

3.2 非建筑物点云提取

我们把地面点剔除之后，可以很明显的看出。剩下的点云由 9 个建筑点云和中间的树木和绿化等点云组成。各个建筑点云和树木和绿化的点云中间都是有一定的距离的，这对于将它们分离有很大的帮助。这里我们选择基于欧式距离进行聚类，将它们各自分离开来。

3.2.1 欧式聚类

欧式聚类，即基于欧式聚类对点云进行聚类。它基于三维点云 k-d 树，快速遍历所有点云，以达到较快速度聚类。具体算法概括为：

1. 找到空间中的某点 P_1，使用 k-d 树找到距离 P_1 最近的 n 个点。判断这 n 个点到点 P_1 的距离，将距离小于阈值 r 的点 $P_{11}, P_{12}, \cdots$ 放入类 Q 里；

2. 在 $Q \setminus P_1$ 中，找到一点找到一点 P_{11}, 使用 k-d 树在全点云中找到距离 P_{11} 最近的 n 个点，判断这 n 个点到点 P_{11} 的距离，将距离小于阈值 r 的非 Q 类中的点 $P_{111}, P_{112}, \cdots$ 放入类 Q 里；

3. 在 $Q \setminus \{P_1, P_{11}\}$ 中，找到一点，重复上述操作；

4. 当类 Q 中不再有新点加入，则完成搜索。

欧式聚类主要可变的参数只有一个，就是搜索邻域允许的最大半径。这个参数也决定了聚类效果。在之后，我们会利用较小的搜索邻域来聚类非建筑点云，因为这样的聚

类更加细致，能够将建筑之外的离散点和建筑之外的小聚类点提取出来，留下大聚类的建筑点云。之后再用较大的搜索邻域来聚类建筑点云，这样就可以实现将非建筑与建筑分离，以及将各建筑之间相互分离。

3.2.2 非建筑物点云剔除

按照上文所说的欧式聚类，并且将搜索邻域半径设置较小，将非建筑物点云一一聚类提取出来，并且提出，结果如下图所示。

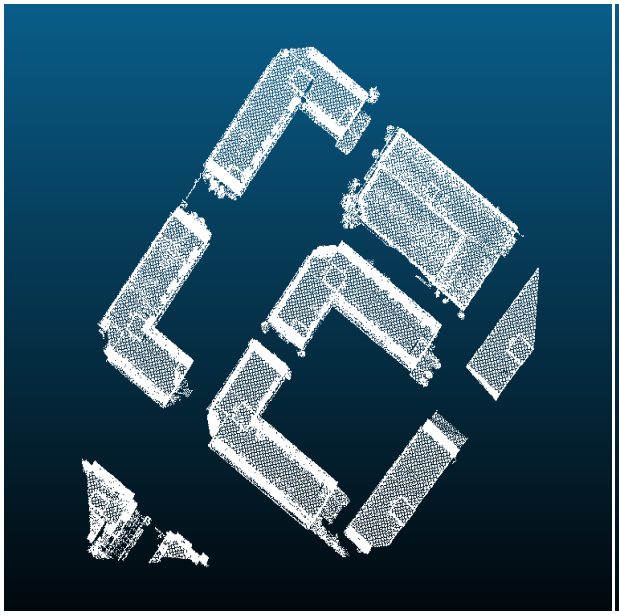


图 3.4 建筑点云

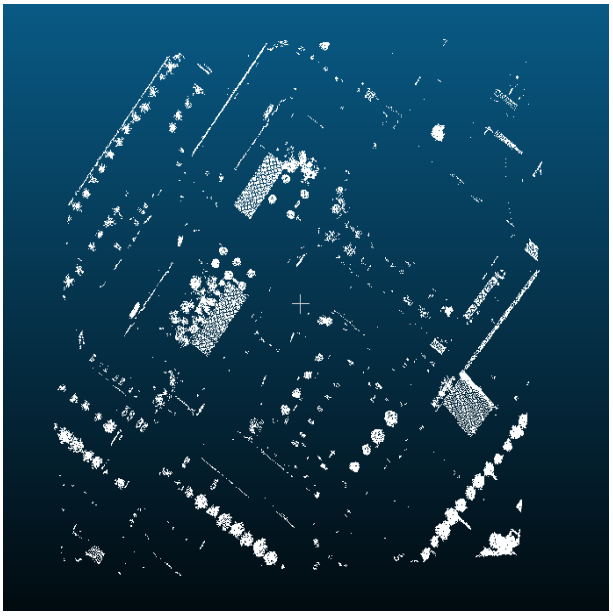


图 3.5 非建筑点云

我们可以看到，9 栋建筑的点云基本已经完全从其他点云中脱离出来了。但是仍有一些残留的与建筑物连接过于紧密的树木点云没有分离出来。这对我们之后的工作提出了一个新的要求，我们需要有其他的方法阻止这些树木点云对模型构建的影响。这在之后将会谈到。

3.3 建筑点云聚类

正如上文所说，在我们得到 9 个建筑点云之后，我们再次使用欧式聚类，并把搜索邻域半径提高。当这个半径大于单个建筑内部点云距离而小于建筑与建筑之间的距离的时候，我们就可以把 9 栋建筑一一单独分离开来。

如下图所示，9 栋建筑被单独分离开来，为了方便表示我们将 9 栋建筑编号，也标在了图中。之后我们选取 9 栋建筑中有代表性的建筑与立面进行接下来的步骤的阐释。

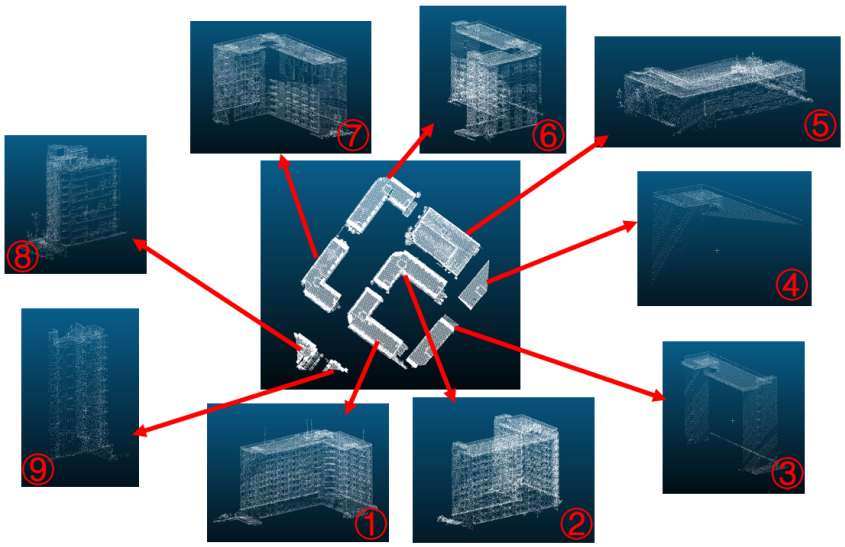


图 3.6 建筑点云聚类结果

3.4 墙面提取

在一般的城市点云模型重建中，这个时候只需要提取出屋顶点云就足够了。但是，因为本文的重点是放在了建筑细节特征提取上，主要也就是门窗提取上。所以在这里，墙面提取也是十分必要的。我们把屋顶提取和墙面提取统一到一起。使用 RANSAC 算法，用平面拟合，将墙面和屋顶同时提取出来。

我们以上图中①号建筑为例，使用 RANSAC 提取平面，不仅提取除了屋顶和 7 个墙面，同时也将屋顶阁楼的顶面也提取了出来，这方便我们之后的模型构建。我们给每个立面表上序号，结果如下图所示。

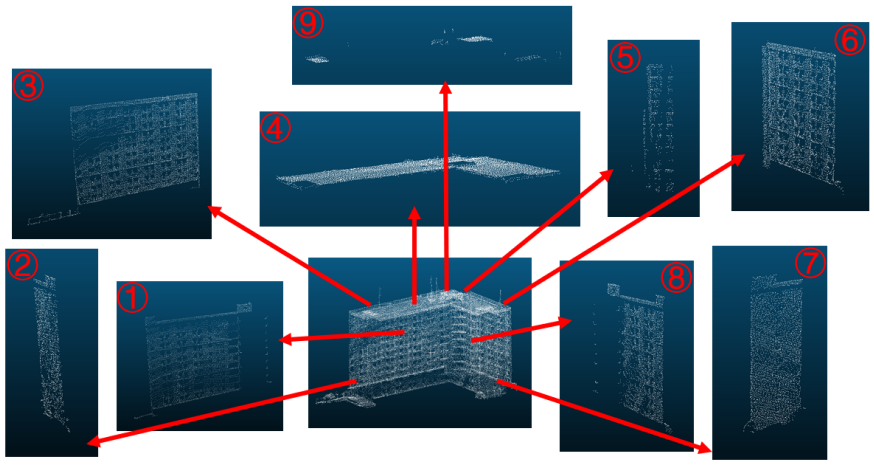


图 3.7 建筑①立面提取结果

同样的方法，我们可以把每一栋建筑都做立面提取。之后的论文会挑选部分立面为例来阐述各个方法。其他建筑与其他立面方法类似。

4 门窗提取

4.1 空洞提取

在绪论部分，我们介绍了当今城市点云模型重建的研究现状。之前的各个学者的研究重点，都在如何能够更快、更清楚的构建出建筑物的基础模型。但是在近几年，人们对于细节部分重建的要求越来越大，其中一个很大的重点，就是对于建筑物上的门窗如何重建，这也正是本文的一大重点。

对于门窗的重建主要有两个关键问题，其一，是如何准确的提取立面上的空洞，其二，是遇到因为某些原因导致 LiDAR 扫描没有扫描到的地方，有没有什么能够通过点云分析做弥补的地方。在之前，我们已经更成功的把每个建筑的每个立面都提取了出来，现在我们就单独拿出几个立面来举例，阐释各个步骤以及相关原理。

首先是空洞提取，关键就是把每个门窗扫描留下的空洞的坐标中心坐标找到。我们以第①号建筑的第①号立面为例，如下图所示。该立面十分完整，每个窗户的空洞都十分明显，非常适合用于试验。

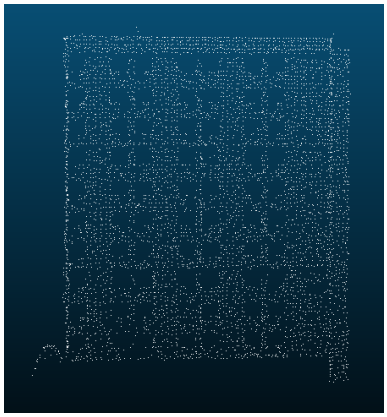


图 4.1 第①号建筑第①号立面

观察可以发现，一个立面上的窗户空洞都是非常具有规律的，一般都是按照矩阵排列，并且各行各列对齐。这就体现了几个关键特征：第一，我们并不需要把每个空洞位置都找到，只需要找到所有空洞的横坐标，和所有的纵坐标，然后构建矩阵就得到了每个空洞的坐标。第二，因为行列对齐，所以如果我们按行或按列做栅格计算每个栅格的点数的话，在空洞的地方会有明显的数量下降。

我们依据此，选择我们使用的方法：对立面做横竖栅格，计算每个栅格中点的数量，

得到一个点数关于纵坐标和横坐标的离散点。我们对这些离散点做多项式插值，提取拟合函数的局部极小值点，也就是每个空洞的中心的横坐标或纵坐标。然后将横纵坐标交叉生成矩阵，也就得到了每个空洞的中心坐标。

假设对某立面做 $n+1$ 横向栅格，记为栅格 $x_0, x_1, \cdots, x_n$。计算每个栅格内的点云数，得到 $f(x_0), f(x_1), \cdots, f(x_n)$。设函数组 $\{\varphi_k(x)(k=0,1,\cdots,n)\}$ 是次数不高于 n 的多项式组，且在 $x_0, x_1, \cdots, x_n$ 上线性无关。

则若在次数不高于 n 的多项式集合 $\mathscr{D}_n = \mathrm{Span}\{\varphi_0, \varphi_1, \cdots, \varphi_n\}$ 中存在多项式

$$p_n(x) = \sum_{k=0}^{n} c_k \varphi_k(x). \tag{4.1}$$

使其满足

$$p_n(x_i) = f(x_i), i = 0, 1, \cdots, n. \tag{4.2}$$

则称 $p_n(x)$ 为满足插值条件的 n 次插值多项式。

我们将求得的 $p_n(x)$ 插值多项式函数细分，提取出其中的局部极小值点，如下图所示。圆圈就是提取出的局部最小值点。

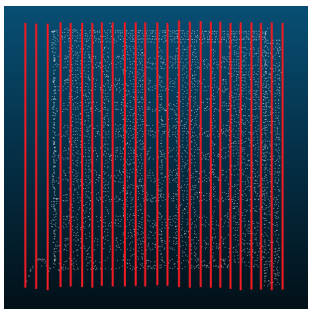


图 4.2 横向栅格

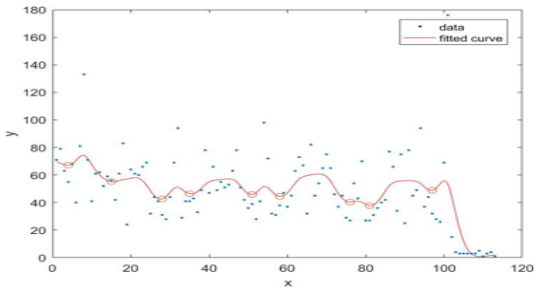


图 4.3 横向插值函数

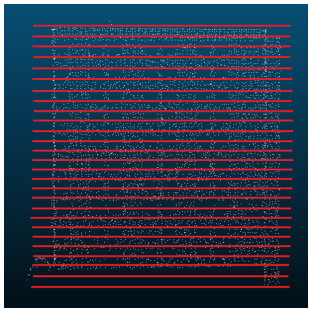


图 4.4 纵向栅格

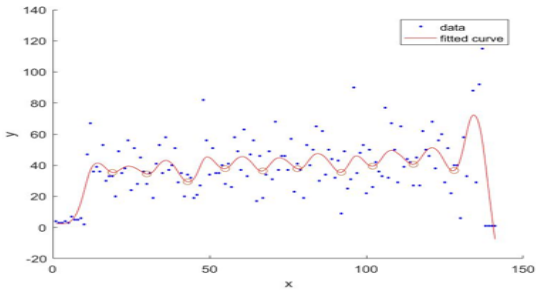


图 4.5 纵向插值函数

这其中有两个就要注意的地方。第一是选取合适的栅格。因为一般的墙面都是竖直的，所以选取横向栅格只需要按照 z 轴划分就可以了。但是横向栅格就不一样了，不合

理的栅格宽度选择对点云数计算非常的不合适。我们需要考虑在前一步 RANSAC 提取立面是拟合的平面的法向量，求出所求栅格宽度在真实 x 轴上的长度，这样划分的栅格才更加合理。

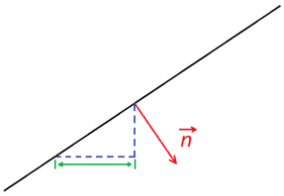


图 4.6　宽度转化示意图

第二个需要注意的地方就是，在拟合函数时求的极小值是局部极小值，并没有什么限制。所以如果出现立面上因为扫描不均匀导致的某一块儿的点云少量缺失，也会导致产生不需要的极小值点。这是需要加以限制来规避的。

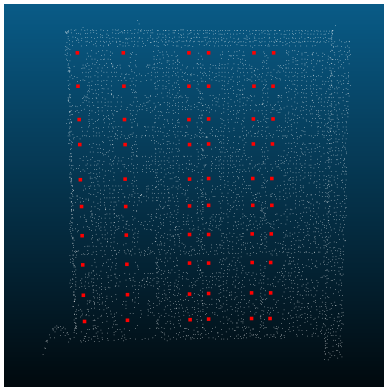


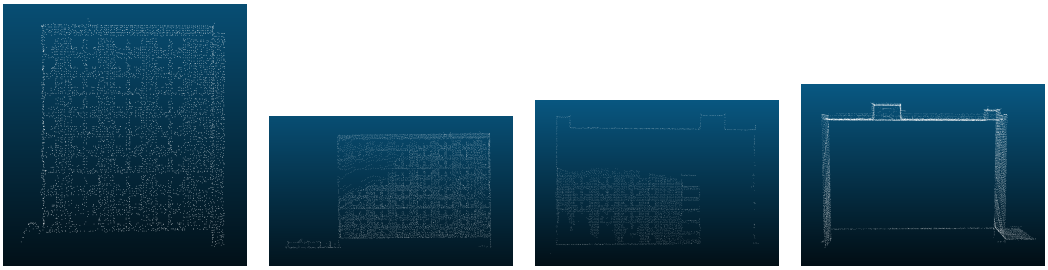
图 4.7　第①号建筑的第①号立面空洞坐标提取

上图就是第①号建筑的第①号立面的空洞坐标提取结果。可以看出来提取的还是较为准确的，基本把空洞的位置都找了出来，之后，我们可以通过空洞的位置构建门窗。

4.2　遮挡填补

在门窗提取的相关研究中，有一个非常重要的问题，就是如果遇到因为树木等物体遮挡住了墙面的部分点云，我们该如何进行门窗重建。

我们可以被遮挡的情况分为以下四类：无遮挡、轻微遮挡、部分遮挡和严重遮挡。



(a) 无遮挡　(b) 轻微遮挡　(c) 部分遮挡　(d) 严重遮挡

图 4.8　遮挡情况

对于无遮挡的立面，正如前文第①号建筑的第①号立面那样，其中的门窗空洞是可以得到很好的提取的。对于轻微遮挡的立面，当遮挡只是对点云密度造成了一个小的波形时，可能会对空洞提取造成一定的干扰，不像无遮挡一样准确，单体大体上前文的提取方法还是依旧可以使用。下图是一个轻微遮挡的立面，使用前文的方法，依旧可以提取空洞的大致位置。

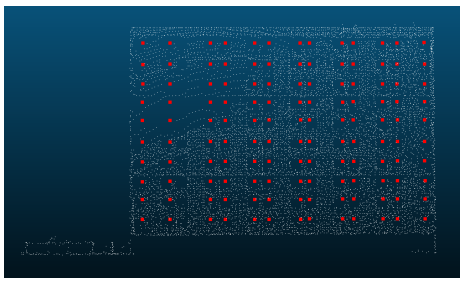


图 4.9　第①号建筑的第②号立面空洞坐标提取

对于部分遮挡的立面，单使用前文的方法就不够了。部分遮挡的立面，一般都只有一部分点云信息完整，而另一部分点云信息缺失。但是，常识告诉我们，在一般情况下，大多数的建筑，各楼层之间是有一定的相似性的。也就是说如果部分楼层的某个位置有一个窗户，那其他楼层的这个地方很可能都有一个窗户。我们可以利用这个一般特征，来对于遮挡的立面部分做位置的估计。

我们以第⑥号建筑的第①号立面为例。对该立面做纵向栅格，计算栅格内点云，做插值函数，可以提取出立面下半部分的局部极小值点。因为我们认为各楼层有相似性。所以我们把两个极小值点之间的距离 h 估计为该栋建筑的楼层高度。那么对于下半部分提取出的空洞坐标，加上一定数量的楼层高度，我们在那里也添加一个门窗。

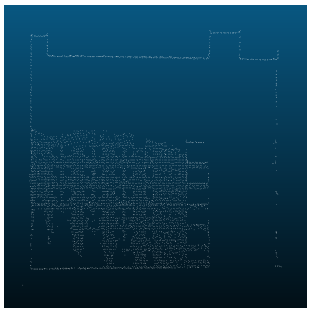


图 4.10　第⑥号建筑的第①号立面

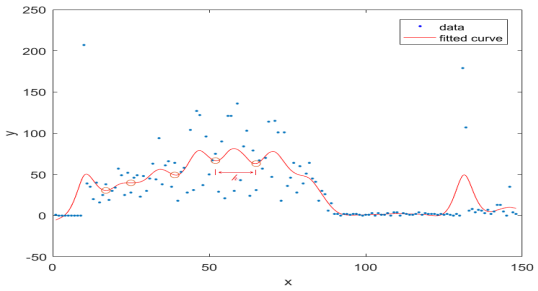


图 4.11　纵向插值函数

下图左就是根据前文方法提取出的第⑥号建筑的第①号立面下半部分的空洞坐标，右图是补全之后的整个立面的空洞坐标。

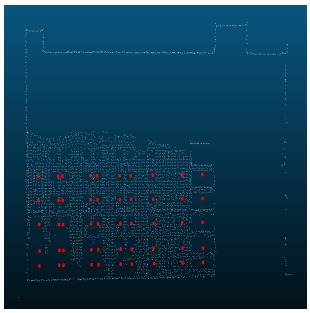


图 4.12　第⑥号建筑的第①号立面空洞坐标提取

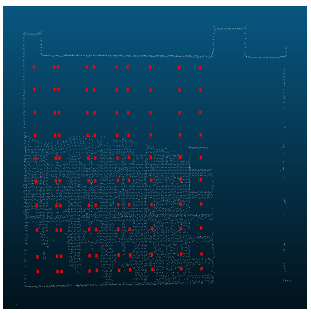


图 4.13　空洞坐标补全

最后，对于严重遮挡的立面，因为缺失信息太多，我们很难在其中提取出门窗的位置，只能放弃。将该立面视为无门窗的全墙壁立面进行重建。

综上，我们就解释完了门窗提取的原理和方法，之后，将该方法运用于各个建筑的各个立面中，就可以依旧完成各立面的门窗提取工作。

5 模型重建

5.1 屋顶轮廓提取

在完成门窗空洞坐标的提取之后，我们就要开始构建房屋模型，然后再把门窗加上去。构建房屋模型有很多种方法，但是因为本文的重点是房屋结构提取，所以我们选用较为简单的模型重构方法，在保持房屋的一定基本信息的基础上加上门窗。

所谓构建模型，就是将模型中的点连接成面，使原来的点云模型构成一个立体的 3D 模型，而在连接成面片的过程中，三角形，一般会作为平面的基本组成单位。将点云用各个三角形连接，然后再给每个三角形着色与渲染，就组成了一个平面或者曲面。这种由三角形组成的平面或者曲面，我们称之为三角面片。

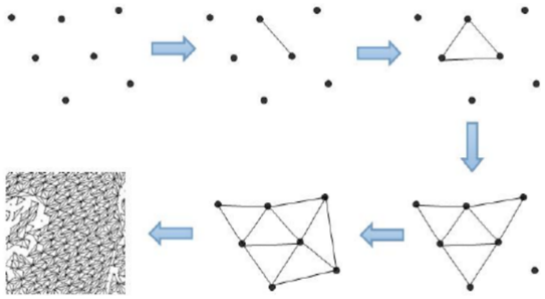


图 5.1 三角面片

为了简化房屋模型，我们一般可以将一栋建筑简化为一个立方体。这个立方体一般由可以由几个长方体组成的。每个长方体除去地面一共 5 个面，每个面作为一个矩形可以拆分成两个三角形。这也就是为什么我们能够将所有的建模模型用三角网格来构建的原理。

如果用立方体来简化每一栋建筑，那么核心问题就是这个立方体的顶面究竟是怎样的多边形，这就需要分析屋顶轮廓的形状。提取屋顶的轮廓，常用的方法有外接多边形法和局部凸壳算法等。我们这里选用局部凸壳算法。

下图是局部凸壳算法的示意图。在前文中，我们已经在提取面片的过程中同时把各个建筑的屋顶立面都已经提取出来了。现在我们把屋顶的点云投影到一个平面上。以下图示意图为例，选取点集最左下角点作为起始轮廓点，以水平向左的方向为起始方向，然后在一定范围内搜索下一个轮廓点，顺指针旋转方向角，取方位角最小的那个点为下一个轮廓点。以此循环，直指回到起始轮廓点。

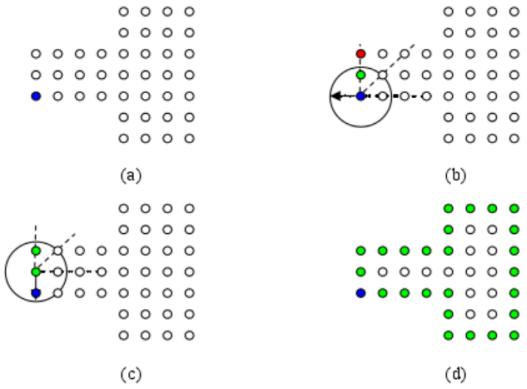


图 5.2 局部凸壳算法示意图

下图就是我们通过上述算法提取出的第①号建筑的屋顶轮廓。因为选用了 PCL 的库，PCL 的函数将内边界点也一并提取了出来。

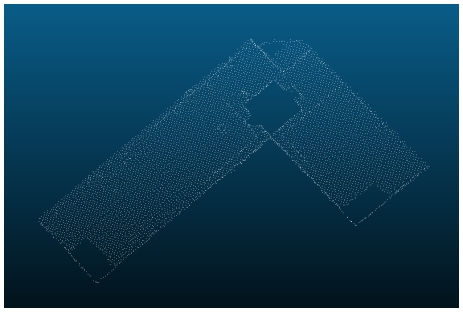


图 5.3 第①建筑屋顶

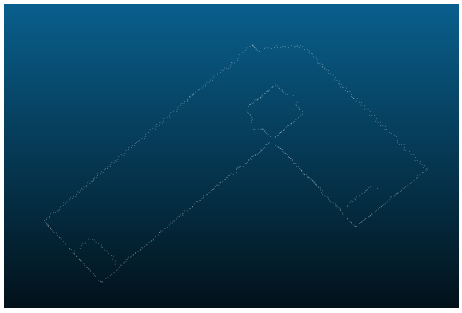


图 5.4 第①建筑屋顶轮廓

5.2 轮廓拐点重建

但提取出屋顶的轮廓还是不够的，为了使建筑模型尽可能简洁，我们需要用尽可能少的三角形去连接屋顶轮廓点，也就是我们要尽可能把屋顶轮廓中所有的拐角点找出来。

当我们把屋顶轮廓视为一个多边形的时候，所谓拐角点，也就是屋顶各条边之间的交点。首先我们需要，将屋顶的轮廓按各条边分离开来。这里仿照前文 RANSAC 算法的思想，对平面点云实行二维 RANSAC 提取直线。

对目标轮廓点云中的任意两点 A 和 B，遍历目标点云中的所有点，对任意一点 P，求空间中点 P 到 AB 所在直线的距离 d。

$$d = \frac{\| \vec{AP} \times \vec{AB} \|}{\| \vec{AB} \|}. \tag{5.1}$$

当距离 d 小于一定阈值 θ 时，我们认为点 P 在 AB 所在直线内。

照此定义，计算所有在 AB 直线内的点数，找寻目标点云中拥有最多点的直线，将这条直线内的所有点提取出来。再在剩下的点云中，找拥有最多点的直线，以此类推，知道剩余的点云明显不构成轮廓中的一条边。

这样，我们就提取出了屋顶轮廓构成的各条边中的点，下面需要用这些点计算出各边交点，也就是轮廓拐点的坐标。

我们取两条边中的各两个点，A、B 和 C、D。尽可能取距离较远的两个点，设直线 AB 和直线 CD 交于点 P。因为可以确定上述五个点都在一个平面内，我们可以把这五个点投影到 $z=0$ 平面上，这时，可以把三维问题转化为一个二维问题。如下图所示。

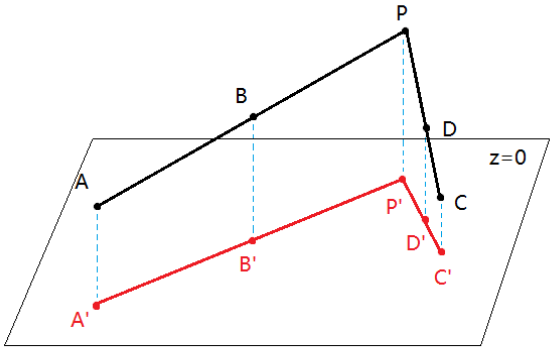


图 5.5　投影求交点

设 $A'(a_1,a_2)$，$B'(b_1,b_2)$，$C'(c_1,c_2)$，$D'(d_1,d_2)$，我们可以求得

$$\frac{A'P'}{A'B'}=\frac{\dfrac{c_2-a_2+\dfrac{b_2-a_2}{b_1-a_1}a_1-\dfrac{d_2-c_2}{d_1-c_1}c_1}{\dfrac{b_2-a_2}{b_1-a_1}-\dfrac{d_2-c_2}{d_1-c_1}}-a_1}{b_1-a_1}. \tag{5.2}$$

根据投影性质，我们有

$$\frac{AP}{AB}=\frac{A'P'}{A'B'} \tag{5.3}$$

据此，我们可以根据点 A 和点 B 的坐标，求出点 P 的坐标。

下图是我们依据上述算法，将第①号建筑屋顶的各个拐点求出来的结果。拐点已用红色的点在图中标出。

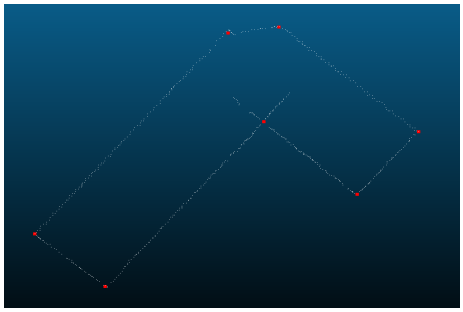


图 5.6　第①号建筑的屋顶拐点

5.3　建筑模型重建

在得到屋顶轮廓拐点之后，我们按照一定的顺序将屋顶拐点连接成为三角面片，然后再将屋顶的点与竖直对应的地面点逐个连接成一个个竖直平面，于是我们就可以得到一栋建筑的简易模型。

下图就是我们构建得到的第①栋建筑的模型。

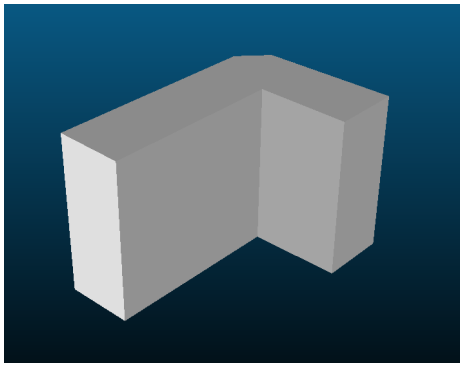


图 5.7　第①号建筑的模型

我们用相同的方法，将九栋建筑模型都重建了之后，再加上地面，就得到了如下图所示的所有建筑的模型。

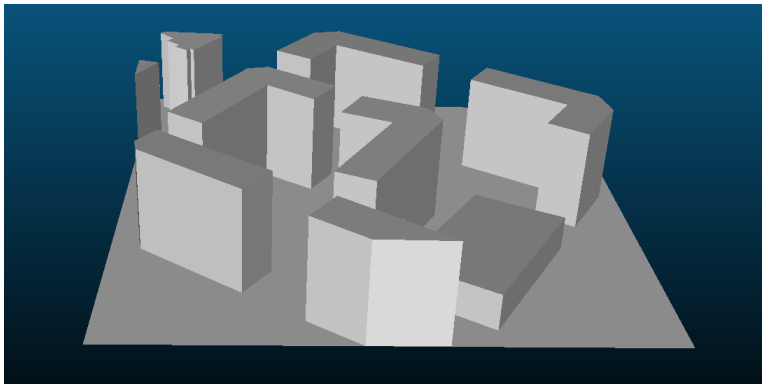


图 5.8　所有建筑的模型

5.4 最终成果

在上文所得到的门窗空洞坐标的基础上，我们依据着墙面的拟合平面，在墙面所在平面上，沿空洞中心坐标周围提取一个长 1.6m 的正方形，作为窗户的模型。比如在第①号建筑第①号墙面上窗户模型构建如下。

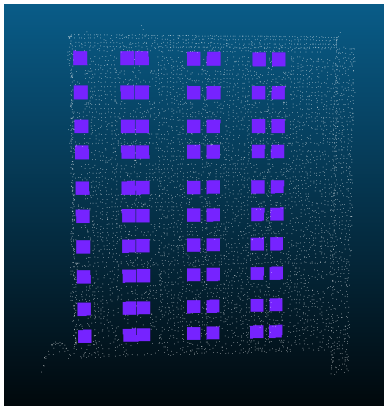


图 5.9 第①号建筑第①号墙面上窗户模型

仿照这个方法，我们把所有的墙面都依据空洞坐标做出窗户模型，然后在加到所有建筑的模型上，就得到了我们的最终成果。

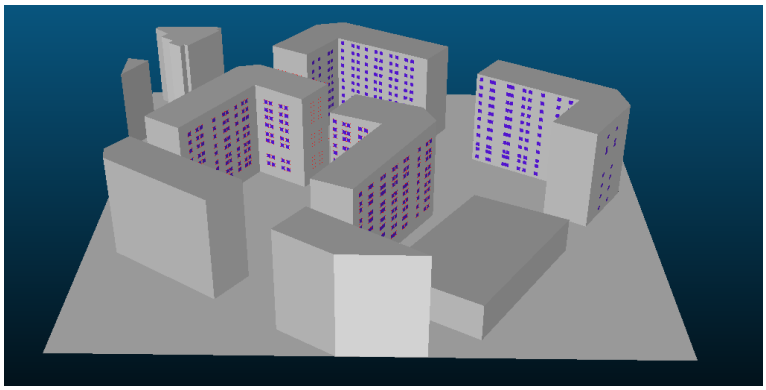


图 5.10 最终模型

结论

至此，我们就完成了城市建模和门窗结构提取与重构的全部过程了。最终的成果也在前文展现了出来。

回顾全文，我们完成了对点云的预处理——包括通过直通滤波和统计滤波对点云去噪、稀疏化与参数化点云，接着我们用处理过后的点云进行了地面点提取、非建筑物聚类与剔除、各个建筑物聚类分离，然后再对着每个建筑物进行每个立面的逐个提取。之后，对于我们提取的竖直立面，也就是墙面，我们进行空洞提取，得到了门窗的中心坐标，而对于我们提取的屋顶立面，我们进行轮廓提取与拐点提取，以构建建筑模型。最后，我们基于得到的建筑顶点坐标和门窗坐标，构建了建筑模型和带窗户的建筑模型。

一方面，本文成果丰硕。我们不仅对每个过程进行了详细的阐释，解释了各个算法的原理，并将其实现。本文主要的成果在于：第一，对点云的预处理从去噪、稀疏化、参数化三个依次进行操作，每一步都做了详细的算法阐释，多方面的处理点云使点云更加适合于之后的建模操作。第二，通过可调节的欧式聚类算法，将非建筑点云与建筑点云分离，并且将建筑与建筑之间分离，使得之后的操作简单可行。第三，通过 RANSAC 算法，将建筑物屋顶及各个墙面点云都提取了出来，即便于构建建筑模型，又可以用于提取门窗空洞。第四，仿照 RANSAC 算法编写二维内的直线 RANSAC 算法，实现了屋顶轮廓拐点的提取，使得建筑模型简洁清晰。第五，利用样条插值，提取墙门门窗空洞坐标，以达到提取门窗的效果，使得建筑模型更加具体。填补了一部分目前城市点云模型重构较为缺乏的内容——将门窗加到建筑模型上。

另一方面，本文也有一定进步的空间。第一，对于建筑的模型过于简单。因为时间限制等原因，我们将建筑物简化为一个立方体，但实际情况下，我们是完全可以将更多的细节构建出来。比如在屋顶可能会有阁楼小方块，这是被我们简化掉的东西，如果加上它们，可以使我们的模型更加准确、逼真。第二，对于门窗的提取不够智能化，考虑到点云的各种复杂情况，很多门窗的提取需要人为调试，这给我们增加了很多的工作量。第三，最后的模型构建渲染效果较为简单，对于一个良好的 3D 模型，应该是兼具准确与美观的，如果技术能力允许，最好能给建筑墙面加上一定的墙面纹路，对于窗户构建一个窗户的模型放在建筑模型上，如果能做到，能够使最终成果的模型更加生动逼真。

除此之外，本文的课题来源主要取自于北航数学与系统科学学院“数学、信息与行为”教育部重点实验室的 2019 年“挑战杯”参赛课题。本文部分算法取自于该课题研究，我与课题组的同学们已共同合作发表了一篇会议论文：《Three-Dimensional Re-

construction Method with Parameter Optimization for Point Cloud Based on Kinect v2》(ICP-MMS2019)，两份软件著作权“基于 Kinect v2.0 的点云获取与模型重建软件”，以及一篇 SCI 论文在投：《SIFT+ICP: An Effective Point Cloud Registration Algorithm for Low Resolution Depth Camera》（IEEE Access）。

总之，本文基本完成了城市点云模型重构的全过程，并且在门窗提取上做出了一定的改进与创新，成功完成毕业设计的预期与目标。

致谢

至此，随着我的毕业论文的全部完成，我的大学四年生涯也就接近尾声了。四年时光匆匆冉冉，却在我的人生中留下来浓墨重彩的一笔。

首先感谢北航，给了我这样的一个成长环境，让我在青春的四年时光里，不仅能够学习到丰富的知识，还能够结识到许许多多志同道合的好朋友。更重要的是，北航给我打开了一片更广大世界的窗户，它让我在四年内接触到了从未想到的大千世界的丰富多彩。它也是我的一个起点，我相信在之后我的求学与工作生涯中，我将会走入一片更广大的天地，在那里，我将永远记住我是从这里出发的。

然后，感谢数学与系统科学学院，学院的老师和工作人员，伴随了我们四年时光，为我们传授了知识，给我们提供了帮助。感谢学院的倾心培养，让我能够成为今天的我。

尤其感谢我的导师——姜鑫老师。结识姜老师是从他的风趣的随机过程课开始，之后又跟随姜老师参加挑战杯，以及如今由他担任我的毕业论文导师，这一切都让我觉得十分荣幸。感谢姜老师对我的毕业论文在选题、方向和进程上提供的帮助。他不仅为我制定了合理的课题与研究方向，并且为我规划好每一步的工作，让我在毕业论文的完成过程中不会觉得迷茫。在除了毕业论文的其他方面，比如挑战杯项目的指导上，他也给我提供了很多帮助，是我在学术前进上的一盏明灯。我十分高兴能够作为姜老师的学生，也祝愿他之后的工作顺利，桃李满天下。

另外，感谢傅锡豪同学在我的毕业论文完成过程中提供的帮助。因为我自己编程能力的缺失，我经常需要向他咨询程序的写法以及 bug 的修复，在其中给他添了不少的麻烦，这让我十分的惭愧。也感谢他能够不怨其烦地教导我，这才促使我能够完成我的毕业论文的全部工作。

最后，感谢我的舍友、我的朋友们大学四年的陪伴，谢谢你们让我的大学生活变得更加有意义。“就算分隔两地，我们看到的仍是同一片天空”。祝愿我的每一个朋友在毕业之后都能够事事顺利，有一个辉煌的前程。

谢谢你们！

参 考 文 献

[1] 张栋. 基于 LiDAR 数据和航空影像的城市房屋三维重建 [D]. 武汉: 武汉大学, 2005.

[2] 喻亮. 基于车载激光扫描数据的地物分类和快速建模技术研究 [D]. 武汉: 武汉大学, 2011.

[3] 魏征. 车载 LiDAR 点云中建筑物的自动识别与立面几何重建 [D]. 武汉: 武汉大学, 2012.

[4] 牛路标. 基于机载和车载 LiDAR 数据的建筑物三维建模方法研究 [D]. 焦作: 河南理工大学, 2016.

[5] 张昌赛. . 布料模拟 LiDAR 数据滤波算法适用性分析 [J]. 激光技术, 2017.

[6] 王庆栋. 基于语义建模框架的机载 LiDAR 点云建筑物三维重建技术研究 [D]. 武汉: 武汉大学, 2017.

[7] G. Sithole G. V. Automatic structure detection in a point-cloud of an urban landscape[A]. 2003 2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas[C]. Berlin, Germany: IEEE, 2003:67–71.

[8] V. Verma S. H. 3D Building Detection and Modeling from Aerial LIDAR Data[A]. IEEE Computer Society Conference on Computer Vision and Pattern Recognition[C]. New York, USA: IEEE, 2006.

[9] A. Golovinskiy T. F. Shape-based Recognition of 3D Point Clouds in Urban Environments[A]. 2009 IEEE 12th International Conference on Computer Vision[C]. Kyoto, Japan: IEEE, 2009:2154–2161.

[10] S. Pu G. V. Knowledge based reconstruction of building models from terrestrial laser scanning data[J]. ISPRS Journal of Photogrammetry and Remote Sensing, 2009, 64(6):575–584.

[11] S. Tuttas U. S. Reconstruction of Rectangular Windows in Multi-looking Oblique View ALS Data[A]. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences[C]. Melbourne, Australia: XXII ISPRS Congress, 2012:317–322.

[12] M. Alexa D. C.-O. S. F. D. L. C. T. S. Computing and rendering point set surfaces[J]. IEEE Transactions on Visualization and Computer Graphics, 2003, 9(1):3–15.

[13] C. Lange K. P. Anisotropic smoothing of point sets[J]. Computer Aided Geometric Design, 2005, 22(7):680–692.

[14] 刘大峰，廖文和，戴宁，程筱胜. 散乱点云去噪算法的研究与实现 [J]. 东南大学学报（自然科学版）, 2007, 37(6):1108–1112.

[15] 刘腾飞. 三维点云数据的预处理研究 [D]. 西安: 西安电子科技大学, 2014.

[16] 刘俊焱，云挺，周宇，薛联凤，李正军. 基于最小二乘的点云叶面拟合算法研究 [J]. 西北林学院学报, 2014, 30(5):70–77.

[17] 卢德唐. 基于移动最小二乘法的曲线曲面拟合 [J]. 工程图学学报, 2004, 25(1):84–89.

[18] M R. C. B. Random Sample Consensus: A Paradigm for Model Fitting with Apphcatlons to Image Analysis and Automated Cartography[J]. Communications of the ACM, 1981, 24(6):381–395.

附录 A 部分关键代码

本论文中的点云操作大多都是基于 PCL 函数库，样条插值的操作是基于 Matlab 软件。这里我将附上本论文中自己编写的部分算法的关键代码。

A.1 RANSAC 算法

A.1.1 平面提取

```
pcl::SACSegmentation<pcl::PointXYZ> seg;
seg.setOptimizeCoefficients (true);
seg.setModelType (pcl::SACMODEL_PLANE);//拟合曲面：平面
seg.setMethodType (pcl::SAC_RANSAC);
seg.setDistanceThreshold (0.05);//拟合平面的距离阈值
seg.setMaxIterations(130000);//最大容量阈值
seg.setInputCloud (cloud.makeShared ());//输入点云
seg.segment (*inliers, *coefficients);//输出点云序号
```

代码 A.1 RANSAC 平面提取

A.1.2 直线提取

```
max_inner = 0;
//遍历所有点云，任取两点
for (size_t i = 0; i < cloud->points.size(); ++i)
{
  for (size_t j = i + 1; j < cloud->points.size(); ++j)
  {
    if (j >= cloud->points.size())
      break;
      //向量 AB
    Inliers->indices.clear();
    AB.clear();
    AB.push_back(cloud->points[j].x - cloud->points[i].x);
```

```
13      AB.push_back(cloud->points[j].y - cloud->points[i].y);
14      AB.push_back(cloud->points[j].z - cloud->points[i].z);
15      Inliers->indices.push_back(i);
16      Inliers->indices.push_back(j);
17      for (size_t k = 0; k < cloud->points.size(); ++k)
18      {
19        if (k == i || k == j)
20          continue;
21            //向量 AP
22        AP.clear();
23        AP.push_back(cloud->points[k].x - cloud->points[i].x);
24        AP.push_back(cloud->points[k].y - cloud->points[i].y);
25        AP.push_back(cloud->points[k].z - cloud->points[i].z);
26        //P 到 AB 所在直线的距离
27        cross.clear();
28        cross.push_back(AP[1] * AB[2] - AP[2] * AB[1]);
29        cross.push_back(AP[2] * AB[0] - AP[0] * AB[2]);
30        cross.push_back(AP[0] * AB[1] - AP[1] * AB[0]);
31        dis = sqrt(cross[0] * cross[0] + cross[1] * cross[1] + cross[2] * cross[2])
              / sqrt(AB[0] * AB[0 + AB[1] * AB[1] + AB[2] * AB[2]);
32        //当距离小于一定阈值时，认为 P 在 AB 直线上
33        if (dis < 0.1)
34        {
35          Inliers->indices.push_back(k);
36        }
37      }
38        //选取点数最大的一条直线输出
39      if (Inliers->indices.size() > max_inner)
40      {
41        max_inner = Inliers->indices.size();
42        res_Inliers->indices.clear();
43        for (size_t t = 0; t < Inliers->indices.size(); ++t)
44        {
45          res_Inliers->indices.push_back(Inliers->indices[t]);
46        }
47      }
```

```
48    }
49 }
```

代码 A.2　RANSAC 直线提取

A.2　求屋顶轮廓拐点

```
1  std::vector<float> A, B, C, D, corner;
2  //选取第一条直线的最远两点
3  float  min = 1000, max = -1000;
4  for (size_t i = 0; i < cloud_1->points.size(); ++i)
5  {
6    if (cloud_1->points[i].x < min)
7    {
8      min = cloud_1->points[i].x;
9      A.clear();
10     A.push_back(cloud_1->points[i].x);
11     A.push_back(cloud_1->points[i].y);
12     A.push_back(cloud_1->points[i].z);
13   }
14   if (cloud_1->points[i].x > max)
15   {
16     max = cloud_1->points[i].x;
17     B.clear();
18     B.push_back(cloud_1->points[i].x);
19     B.push_back(cloud_1->points[i].y);
20     B.push_back(cloud_1->points[i].z);
21   }
22 }
23 //选取第二条直线的最远两点
24 min = 1000;
25 max = -1000;
26 for (size_t i = 0; i < cloud_2->points.size(); ++i)
27 {
28   if (cloud_2->points[i].x < min)
29   {
```

```
30      min = cloud_2->points[i].x;
31      C.clear();
32      C.push_back(cloud_2->points[i].x);
33      C.push_back(cloud_2->points[i].y);
34      C.push_back(cloud_2->points[i].z);
35    }
36    if (cloud_2->points[i].x > max)
37    {
38      max = cloud_2->points[i].x;
39      D.clear();
40      D.push_back(cloud_2->points[i].x);
41      D.push_back(cloud_2->points[i].y);
42      D.push_back(cloud_2->points[i].z);
43    }
44  }
45  //计算比例
46  float lambda;
47  lambda = ((C[1] - A[1] + (B[1] - A[1]) / (B[0] - A[0]) * A[0] - (D[1] - C[1]) / (
        D[0] - C[0]) * C[0]) /((B[1] - A[1]) / (B[0] - A[0]) - (D[1] - C[1]) / (D[0] -
        C[0])) - A[0]) / (B[0] - A[0]);
48  //求出交点坐标
49  corner.push_back(A[0] + lambda * (B[0] - A[0]));
50  corner.push_back(A[1] + lambda * (B[1] - A[1]));
51  corner.push_back(A[2] + lambda * (B[2] - A[2]));
```

代码 A.3　求屋顶轮廓拐点

A.3　样条插值求极小值

```
1  %样条插值函数
2  load enso
3  f = fit(xx,y,'smoothingspline','SmoothingParam',0.05);
4  plot(f,xx,y);
5  plot(xx,y,'.');
6  %求极小值
7  yy=[];
```

```
for i = 1 : length(xx)
    yy = [yy,f(xx(i))];
end
[Pks, Locs] = findpeaks(-yy,xx);
Pks = -Pks;
N = length(Pks);
i = 1;
while (i <= length(Pks))
    if (Pks(i) < 10)
        Locs(i) = [];
        Pks(i) = [];
        i=i-1;
    end
    i=i+1;
end
plot(Locs,Pks,'o');
%计算楼层高度
floor = [];
for i = 2 :length(Locs)
    floor = [floor, Locs(i)-Locs(i-1)];
end
floorh = mean(floor);
%对于部分遮挡的情况，向下向上双向修补
aa = length(Locs);
a = Locs(1) - floorh;
while ( a > 0 )
    Locs = [Locs; a];
    a = a - floorh;
end
a = Locs(aa) + floorh;
while (a < length(y))
    Locs = [Locs; a];
    a = a + floorh;
end
```

代码 A.4　样条插值