湖南大學

HUNAN UNIVERSITY

本科生毕业论文(设计)

设计题目： 基于机器视觉的机械零件三维重建算法设计

学 生 姓 名： 朱峰

学 生 学 号： 201804061429

专 业 班 级： 机自 1808

学 院 名 称： 机械与运载工程学院

指 导 老 师： 陈宁

2022 年 5 月 3 日

湖 南 大 学

毕业设计（论文）原创性声明

本人郑重声明：所呈交的设计（论文）是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

学生签名：  日期：2022 年 5 月 3 日

毕业设计（论文）版权使用授权书

本毕业设计（论文）作者完全了解学校有关保留、使用设计（论文）的规定，同意学校保留并向国家有关部门或机构送交设计（论文）的复印件和电子版，允许设计（论文）被查阅和借阅。本人授权湖南大学可以将本设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本设计（论文）。

本设计（论文）属于

1、保　密□，在_____年解密后适用本授权书。

2、不保密□。

（请在以上相应方框内打“√”）

学生签名：  日期：2022 年 5 月 3 日

导师签名： 日期：2022 年 5 月 3 日

基于机器视觉的机械零件三维重建算法设计

摘　　要

点云数据在机器视觉领域如多视图重建形状检测和分类、形状检测和分类中被广泛的应用。点云采集装置和立体视觉技术仍在不断发展之中，并已经得到了广泛的应用，点云数据处理技术将会是未来最具发展潜力的技术。

点云的配准技术可以应用到机械零件的三维重建模型，在多个视角下对机械零件进行拍摄得到深度图，再将它们逐个配准，直至完成该对象的三维点云模型。在进行三维重构时，将各点集中的各点与其相应点进行匹配，从而达到对齐的目的。

基于深度学习的点云匹配算法日益增多，在 ModelNet40 或 3DMatch 数据集上试验的 PointNetLK、IDAM、PRNet 等网络的速度比传统方法更快，性能也比传统方法更稳定。本设计对配准方法做出了改进，采用了基于 benchmark 网络的深度学习点云配准方法，又结合了 open3d 库中的配准函数进行配准，这两种方法互相弥补缺点，使结果更加精确稳定，误差控制在了 2%左右。

本设计也实现了系统的自动化，程序调用深度相机自动定时拍摄点云图，并自动执行点云处理及配准程序，最终完成对物体的三维重建。考虑对安全、社会、环境、健康等多方面的影响，本设计的方案和设计是可行的。

关键词：点云配准；深度学习；benchmark；三维重建；点云采集

Design of 3D reconstruction algorithm for object based on machine vision

Abstract

Point cloud data is widely used in the field of machine vision, such as multi-view reconstruction shape detection and classification, shape detection and classification.Point cloud acquisition device and stereo vision technology are still developing and have been widely used. Point cloud data processing technology will be the most promising technology in the future.

The point cloud registration technology can be applied to the 3D reconstruction model of mechanical parts. The mechanical parts are photographed from multiple perspectives to obtain depth maps, and then they are registered one by one until the 3D point cloud model of the object is completed.In 3d reconstruction, each point in each point set is matched with its corresponding point, so as to achieve the purpose of alignment.

There are more and more point cloud matching algorithms based on deep learning. The speed of PointNetLK, IDAM, PRNet and other networks tested on ModelNet40 or 3DMatch data sets is faster and the performance is more stable than the traditional methods. This design improves the registration method, adopts the deep learning point cloud registration method based on benchmark network, and combines the registration function in open3d library for registration. These two methods make up for each other's shortcomings, make the results more accurate and stable, and the error is controlled at about 2%.

This design also realized the automation of the system. The program called the depth camera to automatically take point cloud image, and automatically executed the point cloud processing and registration procedures, and finally completed the 3d reconstruction of the object. Considering the impact on safety, society, environment, health and other aspects, the scheme and design of this design are feasible.

Key Words：point cloud registration; deep learning; benchmark; 3d reconstruction; point cloud acquisition

目　　录

插图索引

1 绪论

1.1 课题背景及目的

三维重建技术在小尺寸三维目标的重构和大规模的三维场景重建中得到了广泛的应用，三维重建技术的主要手段是二维和三维点云，点云数据对实际空间的描述要比图像好得多。点云的精细程度会直接影响到点云处理及配准等效果从而最终影响到三维建模的质量，所以点云技术是与三维重构紧密联系在一起的[1]。本课题设计的三维重建算法可以应用到机械零件，建立其三维模型从而可以应用到如物体抓取这种对重建模型的精度要求不太高的场景。目前的三维重建流程主要为：1.获取点云数据：获取光的反射强度、三维坐标和颜色信息等数据；2.点云数据的预处理：比如数据插补和数据精简等操作；3.点云配准：粗配准实现初步的配准。精配准是通过多次收敛迭代，以粗配准为基础进行更高精度的配准，能实现更加精准的结果；4.数据融合：配准后的信息是散乱的，需进行数据融合。5.表面生成：表现出点云的表面，把离散的点云用曲面覆盖。

传感器会产生噪声和离群值，而这些噪声会造成干扰，使点云图无法精确表示出真实物体的形状。于是需要去除离群点；拍摄出的点云图中存在待模型重建的物体以及与其无关的背景，所以需要进行点云分割，删除背景从而只留下要配准物体的点云；点云的形成仅仅是局部重叠，这是因为扫描角度的差异。且如 icp 等传统算法配准的速度或精度较低，在低重叠度或存在大量噪声的情况下很容易无法得到正确的配准结果，而采用深度学习的点云配准方法会较为有效地解决这种问题。此课题设计的目的是将拍摄得到的点云图进行平面删除和离群点删除，并使用深度学习算法进行配准，实验系统实现自动化，自动定时拍摄深度图，并自动完成对深度图的配准；使算法具有较好的鲁棒性和准确性。

1.2 国内外研究状况

1.2.1 点云处理国内外研究现状

由于获取点云的过程中会受到不同因素的干扰，比如被测物体表面的高反射、人为干扰或光照影响、扫描设备分辨率的限制等。这样会使噪声或者离群点等缺陷存在于点云中，而点云处理正是为了修复这些缺陷，提高原始点云的质量，避免后续的表面重建和模型绘制等工作因为点云存在缺陷而受到影响。所以点云处理是相当必要的[2]。

点云降采样：点云降采样是为了降低点云的数量。如 Voxel Grid Downsampling 方法，流程是划分为多个相同的格子，取全部或取随机格子，在这些格子中取一个或随机个点保存。本课题设计采用的即为 Voxel Grid Downsampling 方法，属于下采样。

点云分割：分割点云成不同区域，每个区域有其相似的特征，如纹理特征、几何特征等等。这样做的目的是便于单独处理类似特征的点云，如本设计中的删除平面，就是把点云中含有平面的部分与其他部分分隔开。点云分割方法有很多，例如本设计中的随机抽样一致算法（RANSAC），或其他的比如区域内的生长算法、基于深度学习的分割方法等。

1.2.2 点云配准国内外研究现状

目前点云配准的领域已经有了很多研究成果，例如仿射变换、投影变换等原理，都应用到了很多点云配准的方法中，例如基于几何特征匹配的方法，迭代最近点的方法都是应用比较广泛的[3]。

点云配准可以分为两类：无辅助的自动拼接和有人工辅助的标志点的拼接。无辅助的自动拼接包括四种方法：

（1） 迭代最近点：

提取出点云的局部的信息。缺点是由于物体具有的信息可能比较少，所以可能不是很稳定。

（2） 匹配特征点

利用 SIFT 算法来寻找特征点。

（3）正态分布变换

利用高斯概率分布描述离散点的信息，效率较高且收敛速度快。

（4）依靠平台旋转进行配准

利用旋转平台、摄像机等设备的相对位置、旋转平台的运动等参数，求出各角度旋转矩阵，实现了点云在同一坐标系中的变换，具有较高的准确率，但受到仪器的精度的影响

1.3 课题研究方法

1.3.1 相机标定

相机标定是将外部的世界坐标系统逐渐转化为摄像机内成像平面上的象素坐标系统，从而获得摄像机的模型参数。目前国内外有许多摄像机标定方法，包括常规标定、自标定、主动视觉标定等[4]。利用这些方法，摄像机校准的理论问题已被很好地解决。该项目所使用的摄像机为 Intel RealSense D415 深度相机，标定的方法是使用 Intel RealSense Depth Camera 官网的动态标定软件 Dynamic Calibrator（如图 1.1），标定完成后使用官网的检测软件 Depth.Quality.Tool 进行标定后的检测。检测深度质量主要有三个指标：Z 方向精度（Z-accuracy）、填充率（Fill Rate）、RMS 误差，图 1.2 指出了这些指标。当 Z 方向精度小于等于 2%，填充率大于等于 99%，RMS 误差小于等于 2%时，即可认为标定的质量达到了要求。

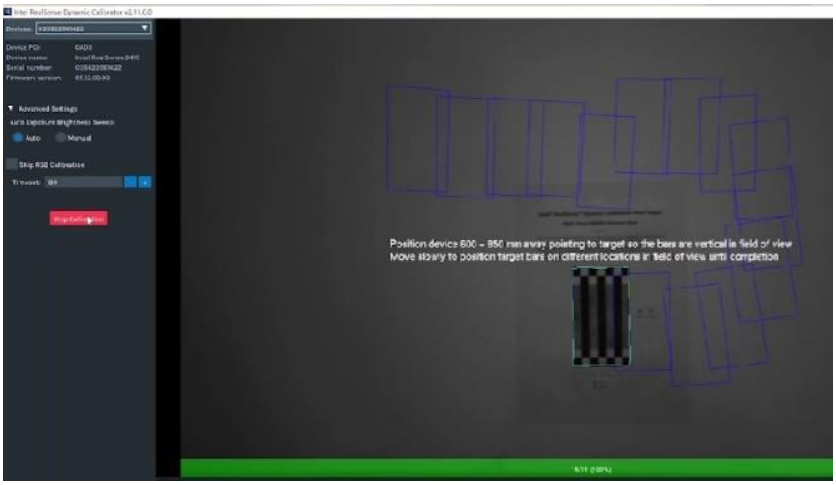

图 1.1 深度相机标定软件

指标	D400/D410/D415 (2米以内，80% FOV)	D420/D430/D435/D435i (2米以内，80% FOV)	D455 (4米以内，80% FOV)
Z方向精度 (Z-Accuracy)	≤ 2%	≤ 2%	≤ 2%
填充率 (Fill Rate)	≥ 99%	≥ 99%	≥ 99%
RMS误差	≤ 2%	≤ 2%	≤ 2%

图 1.2 相机标定指标

1.3.2 点云处理

由深度相机拍摄的某一物体的点云图，点数多，密度大，直接进行点云分割和配准会存在速度缓慢以及精度下降等问题，所以需要先进行点云的降采样，使点云数量减少。

另外由于图中不仅有要重建模型的物体，在其后方还存在背景，此外还可能会存在一些噪声，所以仍然要将点云数据中的噪声即离群点删除掉。

（1） 点云降采样

体素降采样过程分两步：首先将体素网格填充整个点云；然后将每个网格中的点的重心点取代这个体素网格中的所有点，以达到降采样减少点数的目的。体素网格的大小决定着降采样后点数的多少，体素网格尺寸取的大，则最终得到的点数较少。要控制网格大小，取合适的尺寸。若处理后最终点数太少也会无法正常进行配准。

（2）删除平面

为了能使背景较容易地被识别出来并且删除，要把物体放置在平面的背景板前面，以保证拍摄出的点云图只有待建模的物体以及一个或两个平面，这样整个点云就会简单化，便于快速进行点云分割。Open3D 中采用 RANSAC 算法。

（3）删除离群点

本课题设计使用了两种 open3d 库种的删除离群点方法：统计式离群点删除和半径式离群点删除。统计式离群点删除是根据计算出的点云中点云间的平均距离来判断哪些点是离群点，如果大于这个平均距离，则它们就是离群点，会被删除掉。本设计使用 open3d 中的 statistical_outlier_removal 函数，它含有两个参数 std_ratio 和 nb_neighbors。第一个参数设置距离阈值，越小去噪效果越明显；第二个参数设置邻域点数。半径式离群点指定了一个半径，会判读球体周围是否有足够多的邻域点，如果几乎没有，那么会删除这个球内的点。其使用了 radius_outlier_removal 函数。

（4）点云配准

本课题采用了 open3d 库中的点云配准方法与利用 benchmark 网络的深度学习配准方法相结合的方法来进行点云配准。由于传统的配准方法在两个点云的初始位置不好，如重叠度不高时，得到的配准结果会较差；而对于 benchmark 网络，在物体具有复杂结构时不能较好地进行配准，结果也会较差。所以采用先用深度学习配准方法，作为粗配准来调整两点云的位置，使两点云具有较好的位姿，然后使用 open3d 中的点云配准方法，作为精配准进一步优化配准结果。经过这两步的配准，对重叠度不高或初始位置较差的，对具有复杂结构的两点云也可以得到较好的配准结果。

基于 PointNet + Concat + FC 的深度学习配准的 benchmark 方法，结构比较简单，操作比较容易。用来训练和测试的数据集是 ModelNet40，然后应用到深度相机拍摄的深度

图的配准上。

1.4 论文构成

本文按照整个课题设计的流程顺序来介绍各部分内容，从相机标定开始，到点云的预处理，包括点云降采样、点云分割、离群点去除等内容，由于点云配准是本课题的重点部分，所以接下来单独介绍点云配准的原理和方法，最后介绍三维建模的表面生成。

第一章为绪论，首先介绍了课题背景及目的，并介绍了国内外的研究现状，简单列举了一些目前流行的点云处理和点云配准的方法，然后介绍了课题研究的方法，对课题中每一步要解决的问题以及解决的方法做出了简单阐述，最后引出了本论文的构成。

第二章对相机标定的目的、方法和原理做出了详细介绍。

第三章对本课题所用到的点云处理方法逐个进行了介绍，包括点云降采样、点云去除平面背景、去除离群点噪声等。

第四章对本课题用到的点云配准方法进行了介绍，包括 open3d 中的全局配准及局部优化方法及其原理，对 RANSAC 方法进行了讲述，然后介绍了本课题采用的 benchmark 网络的深度学习点云配准方法，包括其网络架构和参数等，然后针对这两种不同配准方法的优缺点进行改进，对两种方法进行改进与综合，使程序能够对不同种类形状的物体都可以得到较好的配准结果。

第五章介绍了三维重建中如何实现点云表面重建以及本设计如何实现自动化。

第六章是本设计的可行性分析。

第七章对本设计做出了分析总结，说明了本文所作的主要工作，总结其优势和创新点，也指出了其缺点和不足。

2 相机标定目的、原理及方法

2.1 相机标定目的及原理简介

计算机视觉要精确地拍摄出三维世界中的物体形状且最终转换成二维图像，必须精确地进行坐标变换，也就是需要相机本身的几何模型参数矫正准确。求出相机的内部参数并且矫正畸变的过程就是相机标定的过程。

要矫正镜头的畸变，才可得到校正后的正常图像，并可以据此来重构三维的场景。相机的校准是做好机器视觉工作的先决条件，这就是其重要性的体现，相机标定的结果直接影响到得到的图像以及后面步骤产生结果的准确性。

三维空间中物体的坐标通过相机成像模型会投影到图像上，应用比较广泛的摄像机成像模型有：拟透视投影、正交投影以及小孔成像模型。而小孔成像模型是其中应用比较多的一种[5]。

相机标定需要经过世界坐标系、相机坐标系、图像坐标系、像素坐标系这一共四个坐标系的转换过程，它们的坐标分别用(X_w, Y_w, Z_w)、（X_c,Y_c,Z_c）、（X,Y）、（u,v）表示，这些坐标系的关系可以用如图的小孔成像模型来表示。通过图中这四个坐标系的关系可以直观的看到，Z_c 轴与 XY 平面垂直，且图像坐标系的左上角顶点是像素坐标系原点（u_0,v_0）。

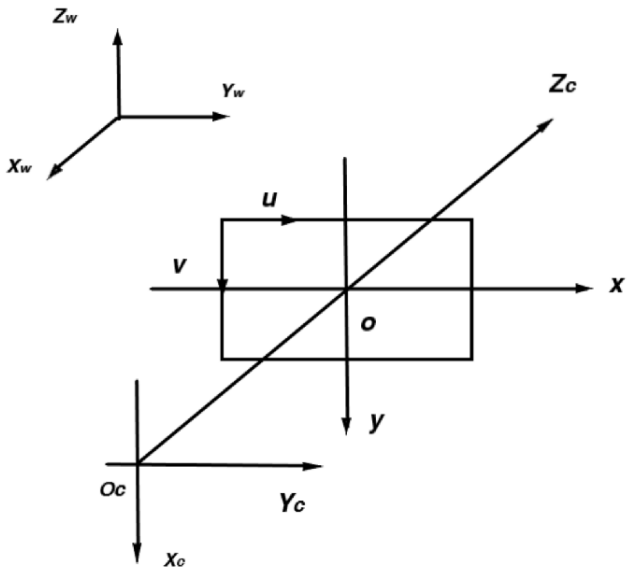

图 2.1 小孔成像模型

校正的首要步骤是将世界坐标系改为相机坐标系，它们的坐标关系如图 2.2，计算变换矩阵过程如下,P 点在相机坐标系中的坐标计算见式 2.1 和式 2.2,其中 R 的维度为 3*3，T 的维度为 3*1。

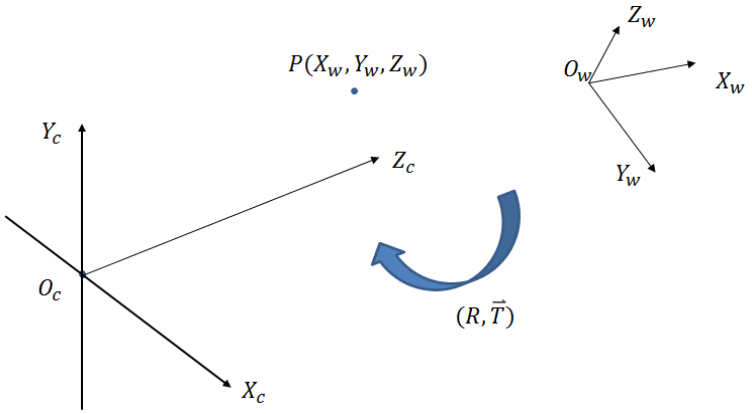

图 2.2 世界坐标系与相机坐标系的关系

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X_w \\ Yw \\ Z_w \end{bmatrix} + T \tag{2.1}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ \vec{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.2}$$

接下来一步是相机坐标系转换为图像坐标系，坐标系的关系见图 2.3，变换矩阵的计算过程见式（2.3）、式（2.4）、式（2.5）、式（2.6）：

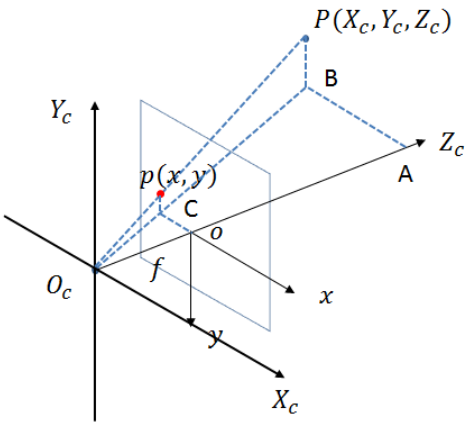

图 2.3 相机坐标系和图像坐标系的关系

$$\frac{AB}{oC}=\frac{AOc}{oOc}=\frac{PB}{pC}=\frac{Xc}{x}=\frac{Zc}{f}=\frac{Y_C}{y} \tag{2.3}$$

$$x=f\frac{xc}{z_c} \tag{2.4}$$

$$y=f\frac{Y_C}{Z_C} \tag{2.5}$$

$$Z_C\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}f&0&0&0\\0&f&0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}X_c\\Y_c\\Z_c\\1\end{bmatrix} \tag{2.6}$$

第三个步骤是将像素坐标系与图像坐标系统之间的变换，见图 2.4，变换矩阵的计算过程见式(2.7)和式(2.8)。

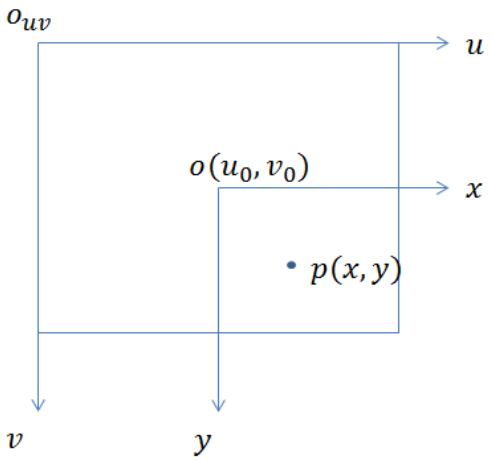

图 2.4 图像坐标系与像素坐标系的关系

$$\begin{cases}u=\frac{x}{dx}+u_0\\v=\frac{y}{dy}+v_0\end{cases} \tag{2.7}$$

$$\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}\frac{1}{dx}&0&u_0\\0&\frac{1}{dy}&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix} \tag{2.8}$$

此外，由于理想的针孔模型与真实的摄像机模型之间的差异，图像会存在径向畸变和切向畸变。径向失真参数由 k1、k2、k3 三个参数来表示，而（x，y）为未校正点的坐标，（$x_{corrected}, y_{corrected}$）是矫正径向畸变后的坐标，其中$r=\sqrt{x^2+y^2}$,见式(2.9)和式(2.10)。

$$x_{corrected}=x(1+k_1r_2+k_2r_4+k_3r_6) \tag{2.9}$$

$$y_{corrected}=y(1+k_1r_2+k_2r_4+k_3r_6) \tag{2.10}$$

用 p_1,p_2 两个参数表示切向畸变参数，见式(2.11)和式(2.12)。

$$x_{corrected}=x+2p_1xy+p_2(r_2+2x_2) \tag{2.11}$$

$$y_{corrected}=y+p_1(r_2+2y_2)+2p_2xy \tag{2.12}$$

所以最终会得到 5 个畸变参数：$D=(k_1,k_2,p_1,p_2,k_3)$将每个坐标系的转换综合起来可以得到一个矩阵，见式(2.13)。

$$Z_c\begin{bmatrix}u\\v\\1\end{bmatrix}\begin{bmatrix}\frac{1}{dx}&0&u_0\\0&\frac{1}{dy}&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}f&0&0&0\\0&f&0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}R&T\\\vec{0}&1\end{bmatrix}\begin{bmatrix}X_w\\Y_w\\Z_w\\1\end{bmatrix}=$$

$$\begin{bmatrix}f_x&0&u_v&0\\0&f_y&v_0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}R&T\\\vec{0}&1\end{bmatrix}\begin{bmatrix}X_w\\Y_w\\Z_w\\1\end{bmatrix} \tag{2.13}$$

2.2 相机标定方法

本课题设计所用的相机是 Intel RealSense Depth Camera D415 深度相机，使用 Intel RealSense Depth Camera 官网的动态标定软件 Dynamic Calibrator 进行标定。

此深度相机是根据立体视觉来计算场景的深度信息的，利用了双目摄像头，这两个摄像头的间隔事实上直接决定了相机的精度和性能。随着时间的推移，相机中的双目摄像头可能会因温度的变化或者撞击而导致微小的位移，导致其间隔或角度发生变化，这就需要进行相机的校准了。采用方法是自校准算法，目前被设计用于校正“内在”或“外在”误差，但不能同时校正两者。内在误差主要是通过透镜位置的微观变化产生的。外部误差与安装两个立体传感器的加强筋的微观弯曲和扭转有关。校准方法是将纹理图放在相机前某个位置，然后将相机放置的足够远，使其超出深度相机的最小范围(aka, MinZ)，但又足够近，使至少 35%的目标纹理在图像中可见。

校准完成后使用官网的检测软件 Depth.Quality.Tool 进行标定后的检测。在检测深度质量主要有三个指标：Z 方向精度（Z-accuracy）、填充率（Fill Rate）、RMS 误差。当 Z 方向精度小于等于 2%，填充率大于等于 99%，RMS 误差小于等于 2%时，即可认为标定的质量达到了要求。

3 点云处理

3.1 点云降采样

3.1.1 点云降采样介绍

由深度相机直接拍摄得到的 pcd 格式或 ply 格式的深度图中会有几万或几十万的点数，直接进行点云分割和配准工作会导致速度缓慢，且配准时会降低配准的精度，导致结果不精确等问题。所以先进行点云降采样十分必要。

本课题设计采用的方法为体素降采样，体素降采样使用了规则的体素网格[6]。其基本原则是将点云分成几个小的立方体，在这些小的立方体区域内，用这些点中的一个重心来表示这些小的立方体中全部点。所以改变网格大小也就改变了最终点数的多少。

降采样的步骤为：

（1）计算点集的最小或最大值；

（2）确定体素网格大小；

（3）计算体素网格的维数；

（4）计算每个点的体素指数；

（5）根据步骤 4 中的索引对点进行排序；

（6）根据质心/随机方法选择排序点。

经本设计的实验表明，经过降采样后的点云的数量在两千个至五千个，程序处理速度较快且配准较为精确。

3.1.2 点云降采样的实现

本课题采用的体素降采样方法中使用了 open3d 中的 voxel_down_sample 函数，输入体素网格尺寸大小即可对 pcd 点云进行降采样，在采样后也用 estimate_normals(o3d.geometry.KDTreeSearchParamHybrid(radius=radius_normal,max_nn=30))函数对刚刚完成降采样的点云估计了法线。图 3.1 为在不同体素网格大小的情况下，降采样后点云的状态，从左到右，体素网格大小依次为 0.003、0.005、0.009。

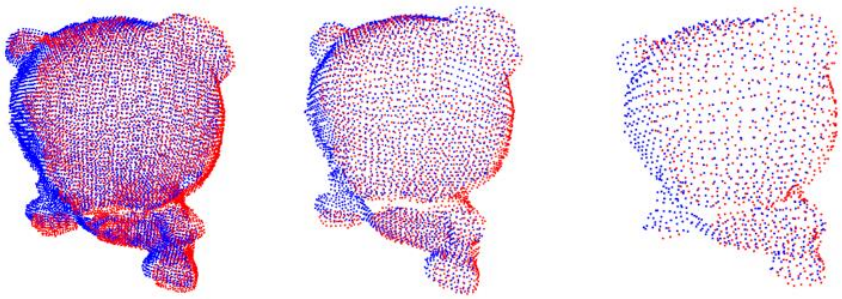

图 3.1 不同体素网格大小对降采样效果的影响

3.2 点云分割

3.2.1 点云分割介绍

点云按空间、几何、纹理等特点进行分割，使得点云具有相似的特性，例如本课题设计中要分割出去的是所拍摄的点云图中的作为背景的平面，如果有多个背景，如两个（或三个）平面，就需要程序循环执行两次（或三次）分割平面的操作。

在对点云进行分类时，必须先对其进行特征辨识，然后利用无语义的低层次属性信息，对某一类点进行分类。“低级属性”是点云数据中可以直接获得的信息然后就可以根据识别的特征进行分割，根据这种低级信息将点分组。

本设计采用的方法是随 机采样一致算法（RANSAC）。RANSAC 算法原理是比较简单的，就是反复不断地进行迭代，并随机地选择某些点。通过这些点来计算模型参数，然后进行判断，把剩下的点代入进这个计算好的模型，来判断是否小于设定误差，如果没有小于，那么就会继续进行迭代。总体来说这个算法有四步：随机采样、计算参数、评估参数、获取最优解。如果某些点云的重叠区域较小或者其中有很多噪声点，也是可以具有较高的鲁棒性，但是缺点是时间复杂度比较高。RANSAC 利用存在离群点的点云，计算出参数。假设性和随机性是 RANSAC 算法主要的思想。它假设了内点（即正确数据），和外点（即错误数据）存在于数据内，由假设性和随机性来计算一些模型的参数。目前 RANSAC 算法在计算机视觉领域中已被广泛应用[7]。

RANSAC 的算法流程：

（1）随机选择两个点。

（2）给出这两个点所表示的数学模型。

（3）在该模型中代入全部数据点并进行误差计算。

（4）选择在误差范围内的点。

然后重复步骤 1~4，迭代多次后找出最适合的模型，即成为算法的解。

3.2.2 点云分割的实现

本课题查找平面的方法使用了 open3d 中的 segement_plane 函数，每次执行函数只能找到一个平面。这个函数具有三个参数：destance_threshold、ransac_n 和 num_iterations。分别代表了为一个点与一个平面之间设置的距离阈值，这个距离上的一个点被称为内点；一个平面的点数；迭代次数。segement_plane 函数执行结束后会输出一个用 ax+by+cz+d=0 表示的平面。本设计中定义了一个 func 函数，输入要去除平面的 pcd 格式点云，使用 segement_plane 函数找到最直观最明显的平面，然后定义 inlier_cloud 为平面内的点，定义 outlier_cloud 为平面外的点。最后 func 函数返回外点 outlier_cloud，即只返回了平面外的点，也就达到了删除平面的目的。

如果有 n 个平面需要去除，就循环执行 n 次 func 函数，最后就会只保留下来需要配准的物体。执行的效果如图，有三个平面，执行三次 func 函数即可把它们全部去除。如图 3.2 展示了本设计中点云删除平面背景的过程。

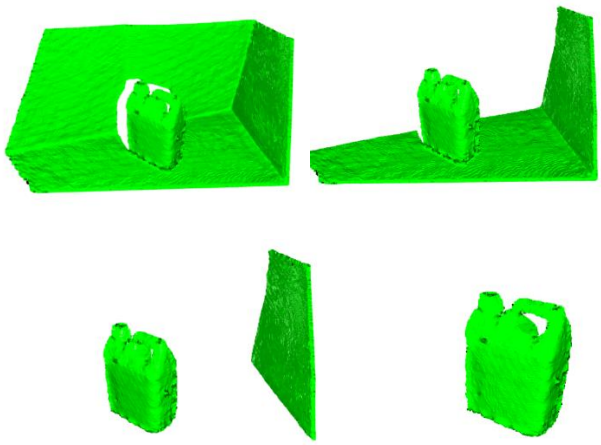

图 3.2 删除平面过程

本设计也进行了点云去噪。为了尽量得到没有噪声的点云，要在尽可能保证其时间空间复杂度不高、体积不变且形状不变的基础上，对原始点云平滑操作。

本设计通过 open3d 两种离群点删除的方法后，深度相机的拍摄中由于光线反射等干扰而产生的噪声点大多数就会被去除掉，避免了无关噪声点影响下一步配准。

4 点云配准

4.1 点云配准介绍

由于扫描设备每次只能从一个固定角度对被测对象进行扫描，所以为了获取物体各个角度完整表面的点云图，需要多次旋转物体，改变物体与相机的相对位置来拍摄点云图。其次需要将这些获取到的每个角度的局部点云拼接对齐，也就是将不同坐标系下的局部点云统一起来，就能得到物体表面完整的三维点云模型。一般来说先进行粗配准，再进行更加精细的精配准。实现配准其实就是一个点集中的所有点与另一个点集中的对应点的匹配过程，它们需要用一个变换矩阵来实现变换。

粗配准常用的方法如下：标签法、基于几何特征的配准法、质心重合法、基于随机采样检测的配准法[8]。

（1）标签法：通过将特征标记标记在对象的表面，来更准确地识别计算各个点云间的变换矩阵来实现配准。方法虽然原理简单，但需要手动去粘贴标志点。

（2）质心重合法：将两个点云的重心计算出来并将它们重合。虽然简单，但缺点是精度低，不是一种经常使用的方法。

（3）基于几何特征的配准方法： 根据点的曲率、积分不变量、特征直方图、旋转图像等几何特征来计算点云的变换矩阵，确定点云相对的位置关系。

（4）随机采样一致性算法（RANSAC）就是不断迭代并进行判断，如果没有足够多的点数在这个误差范围内，那么系统会继续进行迭代，使最终达到有足够的点在误差的阈值内，如果达到了，则停止迭代，结束计算。如果某些点云的重叠区域较小或者其中有很多噪声点，也是可以具有较高的鲁棒性，但是缺点是时间复杂度比较高。

精确配准是让点云间重合部分中的对应点更加精确地对准，是在粗配准基础上进一步调整点的相对位置关系。迭代最近点法（ICP）是用来配准的最常见的方法，其属于精配准。基本思想是：在 Q 和 P 这一对待配准的点云中，寻找点云 P 中与点云 Q 中的点 qi 距离最近的点，这个点是 qi 的对应点 pi。这样它们成为了初始点对(qi,pi)。接下来会进行迭代，算法先计算平移矩阵和旋转矩阵，将此变换矩阵作用到点云 Q，得到点云 Q’并计算与点云 P 的距离，如果式(4.1)表示的目标函数值小于设定误差就停止迭代；或者是到达了迭代次数的设定上限，那么也会停止迭代。否则.就要不断进行迭代直到符合以上两种情况。

$$f(R,T)=\frac{1}{N}\sum_{i=1}^{N}\left\|q_i-\left(R_{p_i}+T\right)\right\|^2 \tag{4.1}$$

配准流程为：

（1）从两点云数据中按相同标准提取关键点 ；

（2）在关键点上计算特征值；

（3）根据两个点云的特征描述子相似性，对位置和特征的对比，对应点对的初步估算；

（4）删除如噪声造成的有错误配对的点对；

（5）用剩余的正确配对的点对来进行变换矩阵的估计。

本设计使用了两种点云配准方法，将深度学习方法和 open3d 配准方法结合起来，增加了配准的精确度，并可以更广泛适应不同形状的物体，下面分别介绍本设计所采用的两种方法。

4.2 0pen3d 点云配准方法

本设计中使用了 open3d 的函数进行配准，在全局配准中使用了 RANSAC 算法，这个算法会用到随机从源点云中选取的点的 FPFH 特征，并进行迭代，用这些 FPFH 特征来寻找源点云的对应点。

其中 RANSACConvergenceCriteria 是一个很关键的超参数。该参数包括两个数值，一个是 RANSAC 算法设定的最大迭代数，一是最大校验次数。这两个数值的大小越大，得到的结果就越精确，但处理速度也越慢。

Open3d 中的配准分为全局配准和局部优化，全局配准代码如图，定义了一个 execute_global_registration 函数，输入两个降采样的点云和它们的 FPFH 特征以及体素网格大小，即可将他们配准。

4.3 benchmark 配准方法

本设计使用的数据集为 ModelNet40 数据集，在训练过程中为了获得点云对，会在样本 template 点云中随机选出 1024 个点，将一个随机产生的平移向量 t 和一个旋转矩阵 R 作用与这些点，使它们平移并旋转到另一个位置而生成 source 点云。另外也在点云对中的所有点加入了随机高斯噪声，这一目的是为增强算法之后对不同情况的对噪声的抗干扰能力。测试时做法与训练时的数据处理方法是大致相同的，只是随机选择 template 点云中的点数为 2048 个点，接下来的操作相同。

本设计使用的网络是 PCRNet，它的体系结构如图，该模型由 5 个 MLPs 组成，大小分别为(64、64、64、128、1024)。源点云和模板点云作为输入通过一组安排在连体架构中的双 mlp 发送。使用一个最大池函数，我们获得全局特征[9]。权重在 mlp 之间共享。这些特性被连接并作为 5 个全连接层 1024、1024、512、512、256 和大小为 7 的输出层的输入提供。前三个输出值表示转换，归一化后的后四个输出值表示旋转四元数。

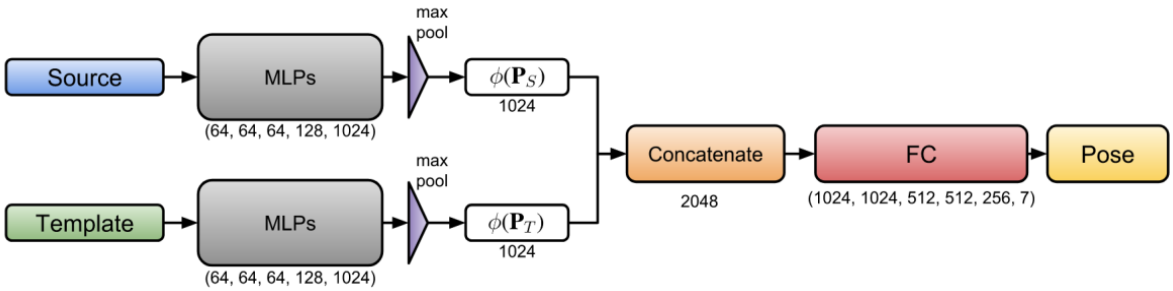

图 4.1 PCRNet 网络结构

但是，由于该网络结构相对简单，通过试验，它无法很好地适应 ModelNet40 的数据，因此对其进行了改进。，设计了 Iterative Benchmark 网络，如图。迭代 PCRNet 架构:迭代 PCRNet 使用图所示的一种改进形式，并不断改进 PCRNet 的估计。首先将模板点云和源点云输入到 PCRNet 中，进行初始偏差 T（1）的预测，这就是第一次迭代。在下一次迭代中，使用 T(1)对源点云进行变换，并将原始模板作为 PCRNet 的输入。在进行 n 次迭代后，我们结合每次迭代的姿态，找到原始源和模板之间的整体转换。

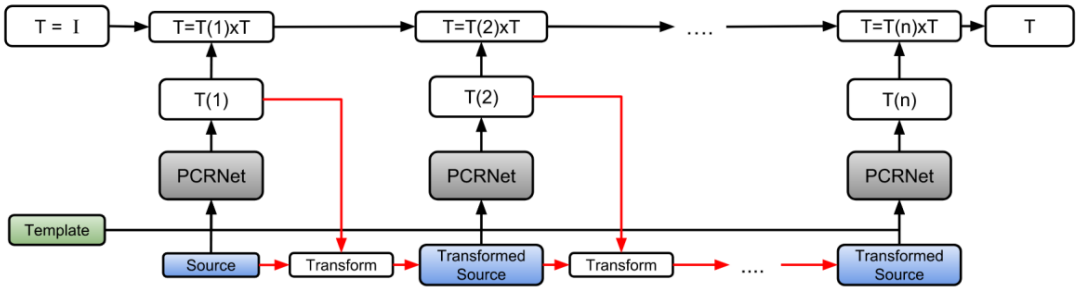

图 4.2 Iterative Benchmark 网络结构

经过 n 次迭代，源点云和模板点云转换是每个迭代转换的结合，见式(4.2):

$$T = T(n) \times T(n-1) \cdots \times T(1) \tag{4.2}$$

本设计先使用 benchmark 的深度学习配准方法，再将 benchmark 方法得到的两点云进行 open3d 全局配准和局部优化，这样结合的方法可以达到较好的精度。

本设计点云配准前后的对比如图 4.3，图中展示了三种物体，每种物体展示了从三种不同角度获得的点云的配准结果，左列红色和绿色的是配准前两点云姿态，右列蓝色和红色的是配准后两点云姿态，可以看到将左边未对齐的点云实现了较精准的对齐。

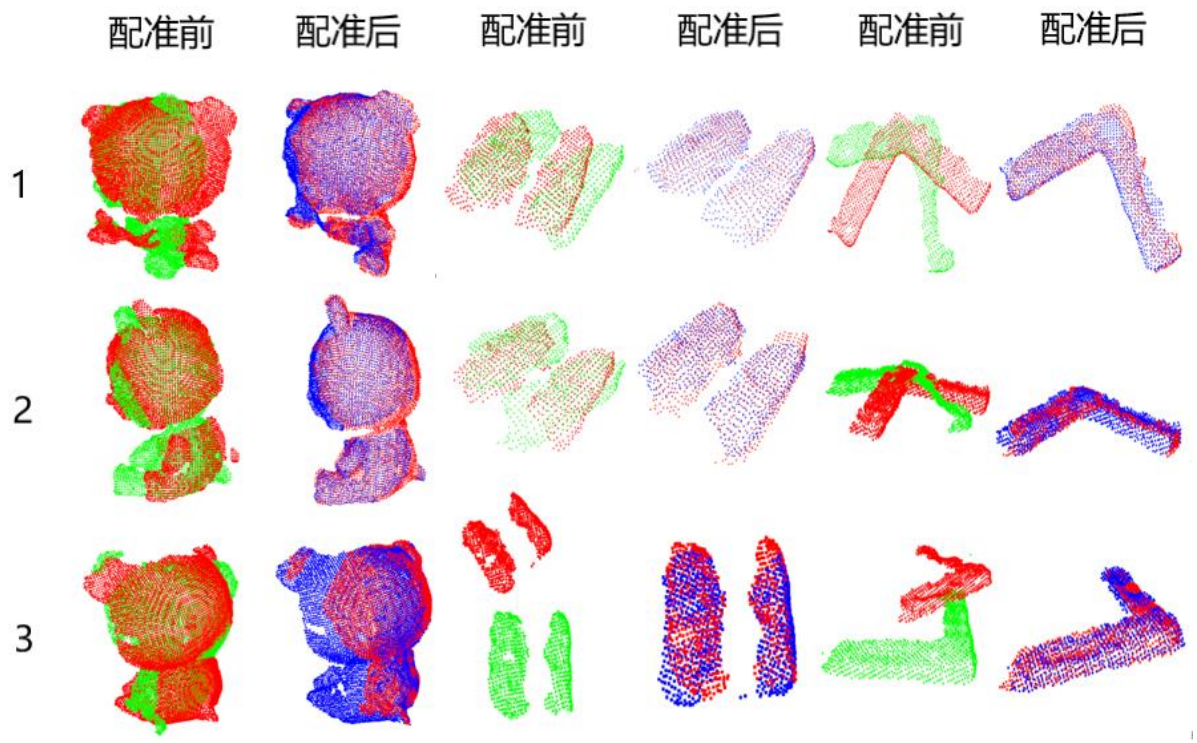

图 4.3 三种物体的点云配准结果

4.4 配准结果精度检测

本设计的三维重建结果可以应用到如物体抓取的工作中，无需较高精度，只需整体各处尺寸与原实物基本一致即可，所以本设计将最后建成的点云模型进行一些关键尺寸的测量，与原实物进行对比，比例一致即可认为重建是成功的。

检验精度的方法为使用 meshlab 软件测量点云模型的各处尺寸，并计算比例。然后测量原物体本身的各处尺寸并计算比例，进行对比。下面是对一个瓶子的三维重建的精度检测过程：

首先测量物体的主要尺寸，物体如图 4.4 所示，并计算它们的比例，得到尺寸为 0.09m、0.11m、0.14m、0.037m、0.07m

图 4.4 待测物体

其次对物体进行三维建模，建模过程省略，如图展示用 meshlab 软件对其测量主要尺寸，依然测量同样的位置，并计算它们的比例。由于物体本身尺寸可能较小，所以在之前的点云处理过程中，为了更加精确地得到配准结果，对点云进行了尺寸放大，如本例对物体尺寸放大了 2 倍，所以在软件中测得的单位也是 m，但尺寸是原物体的 2 倍。如图 4.4 为点云模型的尺寸测量结果，测量尺寸分别为 0.172m,0.223m,0.300m,0.075m,0.138m。

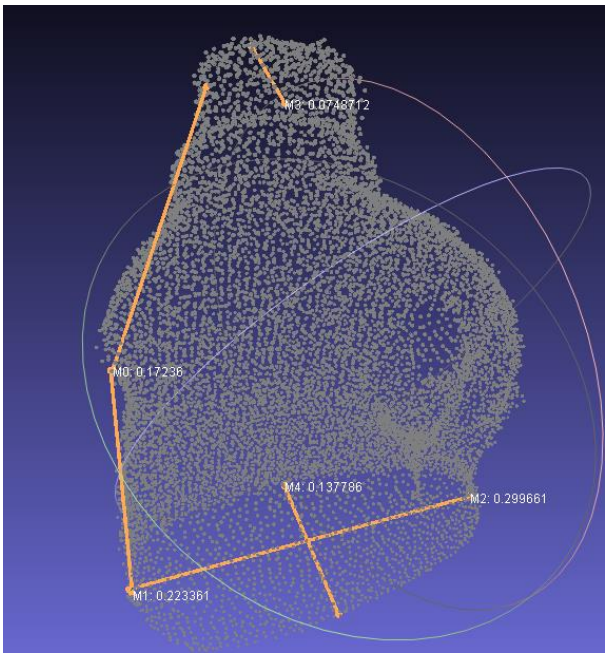

图 4.5 meshlab 软件测量的一些点云主要尺寸

最后进行比较，将它们的对应的尺寸做比：

0.09/0.172=0.523,0.11/0.223=0.493,0.14/0.300=0.47,0.038/0.075=0.507,0.07/0.138=0.507,求得平均值为(0.523+0.493+0.47+0.507+0.507)/5=0.500,标准差为 0.0178,误差小于 2%，对于用于物体抓取目的的工作来说，精度已经达到标准。

5 表面重建与系统自动化的实现

5.1 表面重建简介

目前，关于重构点云表面的再现方法有显式重构和隐重构。显式重构包含了曲面三角形和参数曲面两种不同的方法，即明确地表达目标的位置。隐式重构是一种基于隐式重构的方法，它是用来解决隐含表面的一种方法。另外一种将点集与构造的标量函数的等价平面尽可能接近。常规的曲面重构技术能够有效地解决非均匀性点云的问题，但是很难重构出不规则、杂乱无章的点云。 多边形网格、隐式曲面、参数曲面等都是当前对象模型的主要表现形式[10]。

由于各种绘图软件和硬件技术的发展，使得三角形网格的绘制更加简便。而且，诸如参数或者隐藏的表面等，都要转换成一个三角形的栅格，以便使用原始的绘图。参数化是目前较为常用的一种几何表达方式，但其存在着对复杂几何结构和离散数据进行参数化的缺陷。

5.2 点云表面重建的实现

在 open3d 中，点云重建的方法主要有三种：alpha shape 方法、球旋转算法（BPA）、泊松表面重建法。本设计采用的是 alpha shape 方法，这个方法是先使一个凸包包围住点云，然后逐步向内挖去多余的凸包，显现出点云的表面，并把点云的表面用线和三角形表示，就建成了点云的表面。Open3d 中 create_from_point_cloud_alpha_shape 函数，内部包括一个随意自设定值的 α 参数。这个值取两次，一次是最初始的凸包参数，设置可以较大些，显现出来的形状就是一个包围住整个点云的近似球状的凸包，第二个值是最终的凸包参数，是最终可以表示点云表面的精确的凸包，如图 5.1 为本设计中对完成点云三维建模后的表面重建。当然可以设置凸包向内接近点云表面的次数。

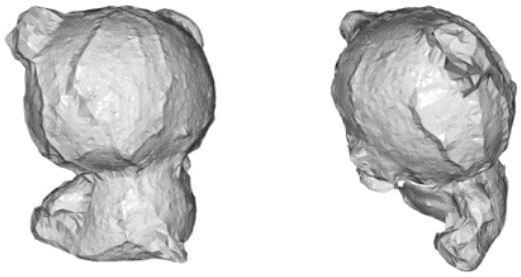

图 5.1 alpha shape 表面重建效果

5.3 深度相机自动获取图像的实现

本设计使用 python 调用 Intel Realsense D415 深度相机进行自动拍摄 ply 格式深度图，并可以转换为 pcd 格式。程序用 frames.get_color_frame()获取了图片的色彩信息，用 frames.get_depth_frame() 得到点云的深度信息，并最后用 o3d.geometry.RGBDImage.create_from_color_and_depth 函数把它们重叠起来，可以得到含有色彩信息的深度图。

要使程序循环拍摄八张照片，需要使程序循环八次。用 for 循环，并每次用 key = cv2.waitKey(1)来间隔 1 秒，即每隔 1 秒拍摄一张深度图，在拍摄间隔期间就可以旋转待建模物体，使下一次拍摄到物体的另一个角度。循环八次后，即得到了物体表面 360°的深度图，这些深度图会自动保存在一个文件夹下。拍摄的效果如图 6.1，左侧为正常的图像，右侧为深度图，颜色的深浅代表距离大小。

图 5.2 python 调用深度相机拍照效果

5.4 点云处理及配准自动化的实现

点云使用 benchmark 及 open3d 配准的程序主要放在 modelnet40_evaluate 子文件中，点云处理及保存文件等程序主要放在 ModelNet40 子文件中。在 modelnet40_evaluate 文件中引用了 ModelNet40 文件。在 pycharm 的终端中输入一段命令即可运行程序。

在 ModelNet40 中，程序依次读取存有拍摄完成的点云图的文件夹中的点云，八个点云经过点云处理后两两配准得到七个点云，这七个点云无需再进行点云处理，直接将它们逐个配准合并得到最终的完整点云。在最初的八个原始点云经过降采样、删除平面和离群点等预处理后，要把它们保存到 h5 文件中，用于 benchmark 方法接下来的配准。由于程序拍摄八张原始点云图，那么需要程序循环 7+6 即 13 次，就可以配准出物体完整的点云模型。

6 可行性分析

6.1 方案可行性分析

本设计采用的点云处理方法为体素降采样与删除平面、删除离群点。若不进行点云降采样，原始点云有大量的点数，配准和删除平面等步骤将花费大量时间进行计算，效率非常低，经过实验，采用本设计的算法，如果几乎不进行降采样而先进行删除平面和离群点，这两个步骤会花费几分钟的时间，且配准也会由于点数太多而导致结果很不精确。而若先进行点云降采样，则仅需要几秒钟就可以完成点云处理与点云配准等步骤。在不同的降采样程度（体素网格大小）下，算法处理时间与效果如图 7.1 所示。

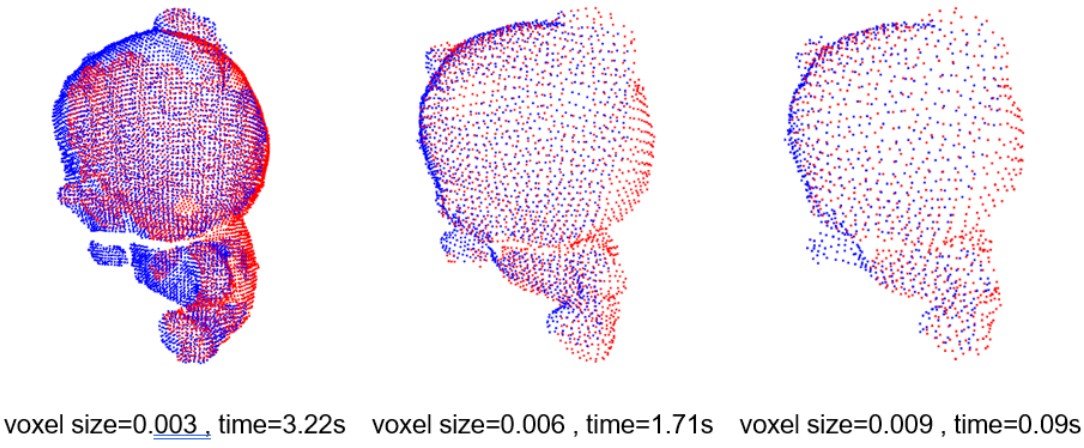

图 6.1 不同降采样程度下处理时间和效果

其次，在点云配准环节，采用了 benchmark 深度学习网络和 open3d 配准相结合的方法，实验证明这两种方法结合是可以弥补它们各自缺点的。图 7.2 最左一列展示了三种物体配准前的位姿状态；第二列展示了仅采用 benchmark 配准方法时，对三种物体的配准结果；第三列展示了仅采用 open3d 的全局配准与局部优化的方法时，对三种物体的配准结果；最右一列是两种方法结合的配准结果。可以看出仅采用 benchmark 方法，对复杂物体的配准效果不是很好，如前两种物体形状复杂，不能很精确地配准，而第三种物体是方盒子，结构简单，可以看出 benchmark 方法配准效果好于前两种；而仅采用 open3d 的配准方法对结构复杂的前两种物体配准效果较精确，但对第三种简单的物体配准却有肉眼可见的误差。所以可见第四列中采用两种方法结合的结果较为精确。

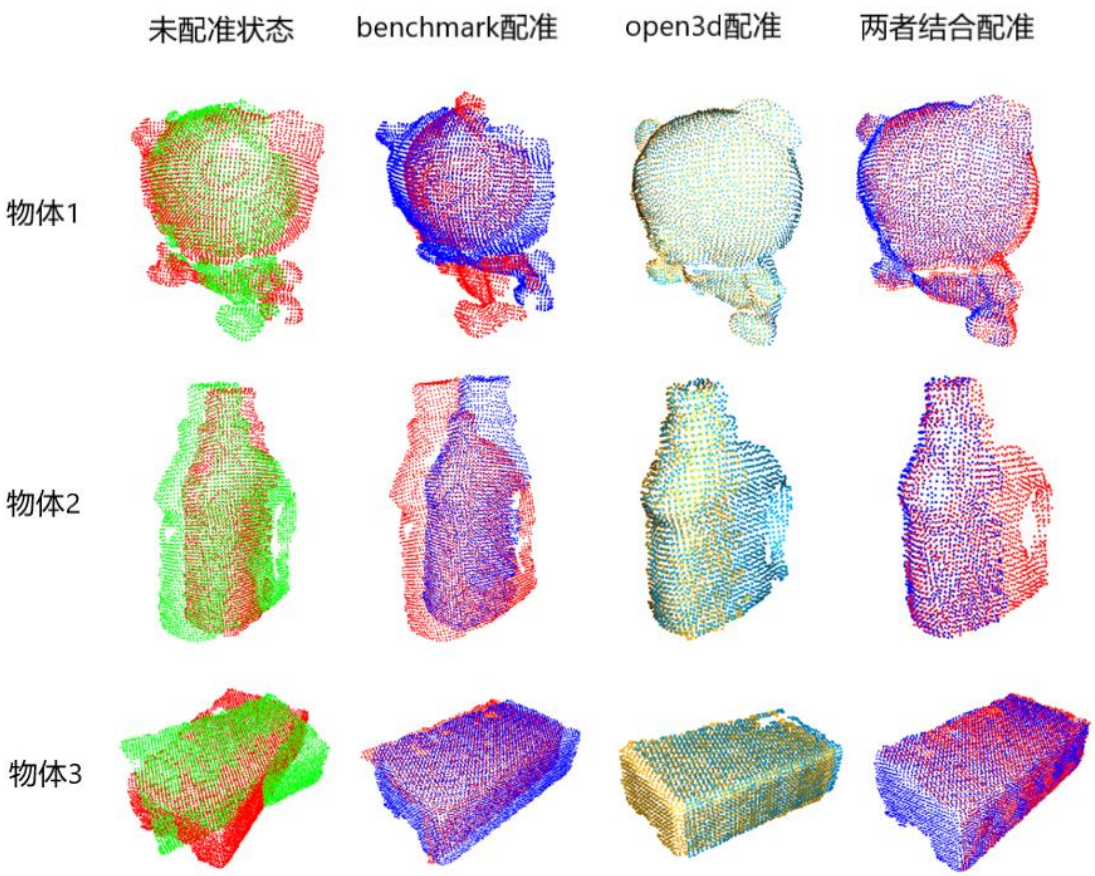

图 6.2 不同配准方法的效果

综上所述，本设计的方案可行。

6.2 技术可行性分析

该方案使用 Intel RealsenseD415 深度照相机作为点云数据采集装置，可处理复杂的深度运算；立体影像感测器能捕捉影像，并能计算影像之间的差别；红外信号发射机能够对目标进行照射并采集深度数据；RGB 感应器能够采集色彩资料。摄像机能产生 90 帧/秒的深度影像，分辨率达到 1280x720，先进的立体深度运算和新的设计，使深度感知更加精确。经过标定后，深度感知的立体视觉误差可以在经过标定后降低为 1%左右。在最佳的条件下，无论是在室内还是户外，摄像机都能捕捉到 10 米范围内的信息。支持 SDK2.0，以及一些编程语言，能被 python 调用，它能自动捕捉对象并获得点云数据，其中包含了色彩和深度。

本设计的点云处理通过 open3d 的各种函数都取得了不错的效果，这些函数在 python 的 open3d 库中，可以很方便地使用。配准步骤中，深度学习 benchmark 配准方法需要

先将拍摄获取到的点云数据进行处理并存入 h5 文件中，然后进行配准。数据集采用了 ModelNet40，其中训练集和测试集的点云数据也是存放在 h5 文件中，这是数据集的一种存放方式。

在最后的点云表面重建中，采用了 α 参数的表面重建法，设置两个参数，也就是建模前后凸包的状态，过程可以理解为逐步把初始的包围点云的凸包向内挖，直到碰到点云的表面，也就自然形成了点云的表面形状。

综上所述，本设计的技术可行。

7 总结

如今，点云三维建模的技术已经有了很明显的进步。本设计其实也叫做三维反求技术，是用三维反求的手段去反求测量某些待测的实物模型，所求得的模型和数据可以用到另外一些工作，如用于设计二维图纸、三维 CAD 数字化的模型等等。这种技术对提升科技水平具有重要意义，且可以使企业响应市场的能力更迅速。考虑对安全、社会、环境、健康等多方面的影响，本设计的方案和设计是可行的。

点云三维模型的研究内容包括：点云数据采集、数据处理以及配准，以及点云表面重建等步骤。本设计也针对这些步骤做出了具体的工作：

（1）点云数据采集：本设计使用 Inter Realsense D415 来扫描物体获得点云深度图，且实现了 python 程序调用相机进行自动拍摄并保存 pcd 与 ply 格式点云，放在同一文件夹下按顺序进行命名，便于后面的自动化处理配准。采集数据前用动态标定软件进行了相机标定并进行了精度检测，以保证获取的点云没有变形，将真实的三维世界中的物体模型表达出来。

（2）点云数据处理：本设计的点云数据处理包括了降采样、点云去噪以及点云分割。由于拍摄得到的原始点云数据具有上万个甚至十几万个点数，处理这样数目巨大的点云不仅仅对算法有很高的要求，效率会降低、速度会很慢，且结果的精确性也不能得到保证。于是采用体素降采样。而点云中由于存在噪声和离群点，经过实验表明，它们会影响配准结果，所以采用了 open3d 中的统计式离群点删除与半径式离群点删除函数来去除噪声；进行平面的分割与去除，并分为内点与外点，输出平面外的点也就达到了删除平面的目的。

（3）点云数据配准：随着深度学习点云配准的方法不断的发展，很多网络在各种数据集上的应用已经取得了不错的效果，所以为了提高点云配准的精确度，弥补传统 icp 算法的缺点，所以本设计采用了深度学习的 benchmark 网络，且将其与 open3d 中的全局配准与局部优化方法结合，取得了不错的效果。既弥补了 icp 的对初始位置敏感的缺陷，也弥补了深度学习网络的对复杂形状物体敏感的缺陷。且本设计实现了自动化配准，将第一步获取到的几个不同角度的局部点云进行自动处理与配准，且将配准结果保存为 pcd 格式与 ply 格式。

（4）点云表面重建：open3d 中的点云表面重建主要有三种方法，本设计使用了 α 参数的算法，将建模完成的点云进行表面重建，使离散的点云数据多边形化为一个完整的

表面，更加直观地表现了物体的形状。

（5）实现系统自动化：本设计对拍摄获取点云数据以及进行后续的点云处理和点云配准的程序实现了自动化，全程用 python 实现，简单易操作。

参考文献

[1]李晋儒,宋成航,林景峰.基于点云数据的隧道曲面三维重建方法的研究[J].城市勘测,2021(06):90-94+99.

[2]秦彩杰,管强.三维点云数据分割研究现状[J].宜宾学院学报,2017,17(06):30-35.

[3]李建微,占家旺.三维点云配准方法研究进展[J].中国图象图形学报,2022,27(02):349-367.

[4]冯亮,谢劲松,李根,霍庆立.摄像机标定的原理与方法综述[J].机械工程师,2016(01):18-20.

[5]郑逢杰,余涛,袁国体,兰进京.相机几何标定方法综述[J].太原科技,2010(02):72-73.

[6]肖正涛,高健,吴东庆,张揽宇.面向三维点云识别的体素网格降采样[J].组合机床与自动化加工技术,2021(11):43-47.

[7]张建伟,权庆乐.一种融合区域生长和 RANSAC 的稠密点云平面分割方法[J].西安科技大学学报,2022,42(02):341-348.

[8]陈亚超,樊彦国,樊博文,禹定峰.基于相对几何不变性的点云粗配准算法研究[J/OL].计算机工程与应用:1-8[2022-04-30]

[9]程诚. 基于深度学习的点云配准关键技术研究[D].南京大学,2021.

[10]陈家兴. 基于特征点的表面重建研究[D].华南理工大学,2020.

[11]宋林霞. 三维点云配准方法的研究[D]. 济南大学.

[12]敖建锋、苏泽锴、刘传立、李美妮、朱滨. 基于点云深度学习的 3D 目标检测[J]. 激光与红外, 2020, 50(10):7.

[13]张匡宇,田庆.一种基于曲率判别的点云去噪与精简算法[J].南方农机,2022,53(03):121-123.

[14]卢文翔,熊瑞平,徐毅松,杨康,李华.基于特征分割识别的点云配准[J].组合机床与自动化加工技术,2022(04):32-35.

[15]荆路,武斌,方锡禄.基于 SIFT 特征点结合 ICP 的点云配准方法[J].激光与红外,2021,51(07):944-950.

[16]秦红星,刘镇涛,谭博元.深度学习刚性点云配准前沿进展[J].中国图象图形学报,2022,27(02):329-348.

[17]夏海明. 点云数据三维表面重建方法的研究[D].哈尔滨理工大学,2010.

[18]李世林,李红军.自适应步长的 Alpha shape 表面重建算法[J].数据采集与处理,2019,34(03):491-499.

[19]刘瑶,张城芳.基于三维激光点云数据的建筑景观可视化建模[J].激光杂志,2022,43(03):138-142.

[20]张睿. 基于深度学习的点云配准研究与应用[D]. 西安电子科技大学.

致 谢

本篇论文落笔之时，也即将到了我顺利毕业之日，鄙人能从毕业设计的选题，到学习基本知识，到完成毕设设计内容，到最终完成论文以及查重，整个过程离不开我的指导老师陈老师与博士周师兄的监督与指导；也十分感谢答辩过程中的几位负责人的老师们，提出了十分宝贵与善良的建议与意见；感谢我的女朋友艺淇，至始至终对我表示支持并提供了不少帮助，没有她在身旁给我加油，我难以全程坚持下来；感谢我的同学：洲洲、dejavu 和 wcx 一直频繁地交流如何修改论文，相濡以沫；以及感谢我的寝室四人的快乐王者荣耀时光，没有几乎每天晚上的开黑活动，大学最后一个学期将会平淡枯燥，没有我们的半夜整活，我的封校时光将会失去光彩，没有了音乐与艺术，没有了生活与摇滚，就没有了人类生存的根本，更何谈毕设；感谢我的父母给予我生命并塑造我的灵魂，感谢机械学院的对我有帮助的老师和同学们，我们会为我们的事业奉献一生，面对我们的骨灰，高尚的人们将洒下热泪！

附录 A　modelnet40_evaluate 配准部分代码：

```
import argparse
import numpy as np
import open3d as o3d
import random
import time
import torch
from torch.utils.data import DataLoader
from tqdm import tqdm
from data import ModelNet40
from models import Benchmark, IterativeBenchmark, icp, fgr
from models.execute_global_registration import g_registration
from models.refine_registration import r_registration
from utils import npy2pcd, pcd2npy
from metrics import compute_metrics, summary_metrics, print_metrics

def evaluate_benchmark(args, test_loader):
    global x
    in_dim = 6 if args.normal else 3
    model = IterativeBenchmark(in_dim=in_dim, niters=args.niters, gn=args.gn)
    if args.cuda:
        model = model.cuda()
        model.load_state_dict(torch.load(args.checkpoint))
    else:
        model.load_state_dict(torch.load(args.checkpoint,map_location=torch.device('cpu')))
    model.eval()

    dura = []
    r_mse, r_mae, t_mse, t_mae, r_isotropic, t_isotropic = [], [], [], [], [], []
    with torch.no_grad():
        for i, (ref_cloud, src_cloud, gtR, gtt) in tqdm(enumerate(test_loader)):
            if args.cuda:
                ref_cloud, src_cloud, gtR, gtt = ref_cloud.cuda(), src_cloud.cuda(), \
                                                 gtR.cuda(), gtt.cuda()
            tic = time.time()
            R, t, pred_ref_cloud = model(src_cloud.permute(0, 2, 1).contiguous(),
                                         ref_cloud.permute(0, 2, 1).contiguous())
            toc = time.time()
            dura.append(toc - tic)
```

```
                cur_r_mse, cur_r_mae, cur_t_mse, cur_t_mae, cur_r_isotropic, \
                cur_t_isotropic = compute_metrics(R, t, gtR, gtt)
                r_mse.append(cur_r_mse)
                r_mae.append(cur_r_mae)
                t_mse.append(cur_t_mse)
                t_mae.append(cur_t_mae)
                r_isotropic.append(cur_r_isotropic.cpu().detach().numpy())
                t_isotropic.append(cur_t_isotropic.cpu().detach().numpy())

                if args.show:
                    ref_cloud = torch.squeeze(ref_cloud).cpu().numpy()
                    src_cloud = torch.squeeze(src_cloud).cpu().numpy()
                    pred_ref_cloud = torch.squeeze(pred_ref_cloud[-1]).cpu().numpy()
                    pcd1 = npy2pcd(ref_cloud, 0)
                    pcd2 = npy2pcd(src_cloud, 1)
                    pcd3 = npy2pcd(pred_ref_cloud, 2)
                    o3d.visualization.draw_geometries([pcd1, pcd3])
                    if x < 8 :
                        voxel_size = 0.004
                        voxel_size1 = 0.004
                    if x >= 8:
                        voxel_size = 0.004
                        voxel_size1 = 0.005
                    if x >= 10:
                        voxel_size = 0.005
                        voxel_size1 = 0.01
                    pcd1 = pcd1.voxel_down_sample(voxel_size)
                    pcd3 = pcd3.voxel_down_sample(voxel_size1)
                    radius_normal = voxel_size * 2
                    print(":: Estimate normal with search radius %.3f." % radius_normal)
                    pcd1.estimate_normals(
                        o3d.geometry.KDTreeSearchParamHybrid(radius=radius_normal,
max_nn=30))
                    pcd3.estimate_normals(
                        o3d.geometry.KDTreeSearchParamHybrid(radius=radius_normal,
max_nn=30))

                    radius_feature = voxel_size * 5
                    print(":: Compute FPFH feature with search radius %.3f." % radius_feature)
                    pcd1_fpfh = o3d.registration.compute_fpfh_feature(
                        pcd1,
                        o3d.geometry.KDTreeSearchParamHybrid(radius=radius_feature,
max_nn=100))
```

```
                    pcd3_fpfh = o3d.registration.compute_fpfh_feature(
                        pcd3,
                        o3d.geometry.KDTreeSearchParamHybrid(radius=radius_feature,
max_nn=100))
                    result_ransac = g_registration(pcd3, pcd1, pcd3_fpfh, pcd1_fpfh,
voxel_size)
                    #pcd3 = pcd3.transform(result_ransac.transformation)
                    #o3d.visualization.draw_geometries([pcd1, pcd3])
                    result_icp = r_registration(pcd3, pcd1, pcd3_fpfh, pcd1_fpfh, voxel_size,
result_ransac)


                    pcd1 = npy2pcd(ref_cloud, 0)
                    pcd3 = npy2pcd(pred_ref_cloud, 2)
                    pcd3 = pcd3.transform(result_icp.transformation)

                    o3d.visualization.draw_geometries([pcd1,pcd3])
                    pcd_combined = o3d.geometry.PointCloud()
                    pcd_combined += pcd1
                    pcd_combined += pcd3

                    if x <= 7 :
                        o3d.io.write_point_cloud(

'C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + str(x) +
str(x+1) + '.pcd',
                                pcd_combined)
                    else :
                        o3d.io.write_point_cloud(

'C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + 'result' +
'.pcd',
                                pcd_combined)


        r_mse, r_mae, t_mse, t_mae, r_isotropic, t_isotropic = \
            summary_metrics(r_mse, r_mae, t_mse, t_mae, r_isotropic, t_isotropic)

        return dura, r_mse, r_mae, t_mse, t_mae, r_isotropic, t_isotropic

if __name__ == '__main__':
        for x in range(1, 14):
            seed = 222
```

```
random.seed(seed)
np.random.seed(seed)

args = config_params()
print(args)

test_set = ModelNet40(root=args.root,
                      npts=args.infer_npts,
                      train=False,
                      normal=args.normal,
                      mode=args.mode)
test_loader = DataLoader(test_set, batch_size=16, shuffle=False)

if args.method == 'benchmark':
    dura, r_mse, r_mae, t_mse, t_mae, r_isotropic, t_isotropic = \
        evaluate_benchmark(args, test_loader)
    print_metrics(args.method,
                  dura, r_mse, r_mae, t_mse, t_mae, r_isotropic, t_isotropic)
elif args.method == 'icp':
    dura, r_mse, r_mae, t_mse, t_mae, r_isotropic, t_isotropic = \
        evaluate_icp(args, test_loader)
    print_metrics(args.method, dura, r_mse, r_mae, t_mse, t_mae, r_isotropic,
                  t_isotropic)
elif args.method == 'fgr':
    dura, r_mse, r_mae, t_mse, t_mae, r_isotropic, t_isotropic = \
        evaluate_fgr(args, test_loader)
    print_metrics(args.method, dura, r_mse, r_mae, t_mse, t_mae, r_isotropic,
                  t_isotropic)
elif args.method == 'bm_icp':
    dura, r_mse, r_mae, t_mse, t_mae, r_isotropic, t_isotropic = \
        evaluate_benchmark_icp(args, test_loader)
    print_metrics(args.method, dura, r_mse, r_mae, t_mse, t_mae, r_isotropic,
                  t_isotropic)
else:
    raise ValueError
```

附录 B　ModelNet40 点云处理部分代码：

```
import h5py
import numpy as np
import open3d as o3d
import os
import torch
from open3d import *
from torch.utils.data import Dataset
import sys
import random

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
ROOR_DIR = os.path.dirname(BASE_DIR)
sys.path.append(ROOR_DIR)
from utils import pc_normalize, random_select_points, shift_point_cloud, \
    jitter_point_cloud, generate_random_rotation_matrix, \
    generate_random_tranlation_vector, transform, random_crop
x = 0
def npy2pcd(npy, ind=-1):
    colors = [[1.0, 0, 0],
              [0, 1.0, 0],
              [0, 0, 1.0]]
    color = colors[ind] if ind < 3 else [random.random() for _ in range(3)]
    pcd = o3d.geometry.PointCloud()
    pcd.points = o3d.utility.Vector3dVector(npy)
    if ind >= 0:
        pcd.paint_uniform_color(color)
    return pcd
#改变点云尺寸和位置函数
def change_scale(pcd):
    points = np.asarray(pcd.points)
    data = np.zeros((points.shape[0], points.shape[1]))
    data = points
    #centre
    xyz_min = np.min(data[:, 0:3], axis=0)
    xyz_max = np.max(data[:, 0:3], axis=0)
    xyz_move = xyz_min+(xyz_max-xyz_min)/2
    data[:, 0:3] = data[:, 0:3]-xyz_move
    #scale
    data[:, 0:3] = data[:, 0:3] * 2
    return data
```

```
#删除平面函数
def func(pcd):
    pcd.paint_uniform_color([0, 1, 0])
    #o3d.visualization.draw_geometries([pcd])
    # 平面分割
    plane_model, inliers = pcd.segment_plane(distance_threshold=0.005, ransac_n=3,
num_iterations=1000)
    # 模型参数
    [a, b, c, d] = plane_model
    print(f"Plane equation: {a:.2f}x + {b:.2f}y + {c:.2f}z + {d:.2f} = 0")
    # 平面内的点
    inlier_cloud = pcd.select_by_index(inliers)
    inlier_cloud.paint_uniform_color([1.0, 0, 0])
    # 平面外的点
    outlier_cloud = pcd.select_by_index(inliers, invert=True)
    # 可视化
    #o3d.visualization.draw_geometries([inlier_cloud, outlier_cloud])
    #o3d.visualization.draw_geometries([outlier_cloud])
    return outlier_cloud

#删除离群点
def outlier_removal(pcd):
    cl, ind = pcd.remove_statistical_outlier(nb_neighbors=20, std_ratio=2.0)
    pcd = pcd.select_by_index(ind)
    cl, ind = pcd.remove_radius_outlier(nb_points=40, radius=0.1)
    pcd = pcd.select_by_index(ind)
    return pcd

#降采样
def preprocess_point_cloud(pcd, voxel_size):
    print(":: Downsample with a voxel size %.3f." % voxel_size)
    pcd_down = pcd.voxel_down_sample(voxel_size)

    radius_normal = voxel_size * 2
    print(":: Estimate normal with search radius %.3f." % radius_normal)
    pcd_down.estimate_normals(
        o3d.geometry.KDTreeSearchParamHybrid(radius=radius_normal, max_nn=30))

    radius_feature = voxel_size * 5
    print(":: Compute FPFH feature with search radius %.3f." % radius_feature)
    pcd_fpfh = o3d.registration.compute_fpfh_feature(
        pcd_down,
        o3d.geometry.KDTreeSearchParamHybrid(radius=radius_feature, max_nn=100))
```

```
    return pcd_down
#输入点云并删除平面离群点
def prepare_dataset(voxel_size):
    print(":: Load two point clouds and disturb initial pose.")
    #输入点云并删除平面
    #ply 转换为 pcd
    global x
    x = x + 1
    ply1 = o3d.io.read_point_cloud('C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + str(x) + '.ply' )

    o3d.io.write_point_cloud('C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + str(x) + '.pcd', ply1)
    source = o3d.io.read_point_cloud('C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + str(x) + '.pcd')
    source = preprocess_point_cloud(source, voxel_size)
    source = outlier_removal(source)
    source = func(source)
    source = func(source)
    source = func(source)
    source = outlier_removal(source)
    #离群点删除
    ply2 = o3d.io.read_point_cloud('C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + str(x+1) + '.ply')

    o3d.io.write_point_cloud('C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + str(x+1) + '.pcd' , ply2)
    target = o3d.io.read_point_cloud('C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + str(x+1) + '.pcd')
    target = preprocess_point_cloud(target, voxel_size)
    target = outlier_removal(target)
    target = func(target)
    target = func(target)
    target = func(target)
    #  离群点删除
    target = outlier_removal(target)
    #降采样
    #source = preprocess_point_cloud(source, voxel_size)
```

```
    #降采样后调整点云尺寸和位置
    source = change_scale(source)
    source = npy2pcd(source, 0)

o3d.io.write_point_cloud("C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/b.pcd", source)
    #降采样
    #target = preprocess_point_cloud(target, voxel_size)
    #  降采样后调整点云尺寸和位置
    target = change_scale(target)
    target = npy2pcd(target, 1)

o3d.io.write_point_cloud("C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/a.pcd", target)
    #o3d.visualization.draw_geometries([source, target])


def read_label():
    filename = 'C:/Users/55276/Desktop/icp/modelnet40_ply_hdf5_2048/ply_data_test1.h5'
    f = h5py.File(filename, 'r')
    label = f['label'][0]
    label1 = f['label'][0]
    return label, label1

def read_data():
    for i in range(0, 1):
        global x
        if x <= 7 :
            path = 'C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/a.pcd'
        else :
            path = 'C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + str(x-7) + str(x-6) + '.pcd'
        pcd = o3d.io.read_point_cloud(path)
        points = np.asarray(pcd.points)
        data = np.zeros((1, points.shape[0], points.shape[1]))
        data[i-0] = points
        print(data.shape)
    for i in range(0, 1):
        if x <= 7 :
            path = 'C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/b.pcd'
```

```
            elif x == 8 :
                path = 'C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + str(x - 6) + str(x - 5) + '.pcd'
                x = x + 2
            elif x > 9 :
                path = 'C:/Users/55276/Desktop/camera/intel_realsense_D415_photo/shendu/lingjian3/' + 'result' + '.pcd'
                x = x + 1
            pcd = o3d.io.read_point_cloud(path)
            points = np.asarray(pcd.points)
            data1 = np.zeros((1, points.shape[0], points.shape[1]))
            data1[i-0] = points
            print(data1.shape)
            if x == 7 :
                x = 8


    return data, data1


def wh5():
    data, data1 = read_data()
    label, label1 = read_label()
    #normal, normal1 = read_normal()
    #faceId,faceId1 = read_faceId()
    f = h5py.File('C:/Users/55276/Desktop/icp/modelnet40_ply_hdf5_2048/dianyun.h5', 'w')
# 创建一个 h5 文件，文件指针是 f'
    f1 = h5py.File('C:/Users/55276/Desktop/icp/modelnet40_ply_hdf5_2048/dianyun1.h5', 'w')
    data.resize()
    data1.resize()
    f['data'], f1['data'] = data, data1   # 将数据写入文件的主键 data 下面
    f['label'], f1['label'] = label, label1   # 将数据写入文件的主键 labels 下面
    #f['normal'] = normal
    #f['faceId'] = faceId
    f.close()
    f1.close()
    print(x)

class ModelNet40(Dataset):
    def __init__(self, root, npts, train=True, normal=False, mode='clean'):
```

```
        super(ModelNet40, self).__init__()
        self.npts = npts
        self.train = train
        self.normal = normal
        self.mode = mode
        files = [os.path.join(root, 'ply_data_train{}.h5'.format(i))
                 for i in range(5)]
        if not train:
            #files = [os.path.join(root, 'dianyun.h5'.format(i))
            #         for i in range(1)]
            if x <= 7 :
                voxel_size = 0.0025
                prepare_dataset(voxel_size)
                wh5()
            else :
                wh5()
            files = [os.path.join(root, 'dianyun.h5'.format(i))
                     for i in range(1)]
            files1 = [os.path.join(root, 'dianyun1.h5'.format(i))
                      for i in range(1)]
        self.data, self.labels = self.decode_h5(files)
        self.data1, self.labels1 = self.decode_h51(files1)

    def decode_h5(self, files):
        points, normal, label = [], [], []
        for file in files:
            f = h5py.File(file, 'r')
            cur_points = f['data'][:].astype(np.float32)
            #cur_normal = f['normal'][:].astype(np.float32)
            cur_label = f['label'][:].astype(np.float32)
            points.append(cur_points)
            #normal.append(cur_normal)
            label.append(cur_label)
        points = np.concatenate(points, axis=0)
        #normal = np.concatenate(normal, axis=0)
        #data = np.concatenate([points, normal], axis=-1).astype(np.float32)
        data = points
        label = np.concatenate(label, axis=0)
        return data, label

    def decode_h51(self, files1):
        points1, normal1, label1 = [], [], []
        for file in files1:
```

```
            f = h5py.File(file, 'r')
            cur_points = f['data'][:].astype(np.float32)
            #cur_normal = f['normal'][:].astype(np.float32)
            cur_label = f['label'][:].astype(np.float32)
            points1.append(cur_points)
            #normal1.append(cur_normal)
            label1.append(cur_label)
        points1 = np.concatenate(points1, axis=0)
        #normal1 = np.concatenate(normal1, axis=0)
        #data1 = np.concatenate([points1, normal1],axis=-1).astype(np.float32)
        data1 = points1
        label1 = np.concatenate(label1, axis=0)
        return data1, label1

    def compose(self, mode, item):
        ref_cloud = self.data[item, ...]
        src_cloud = self.data1[item, ...]
        R, t = generate_random_rotation_matrix(), generate_random_tranlation_vector()
        if mode == 'clean':
            ref_cloud = random_select_points(ref_cloud, m=self.npts)
            #src_cloud_points = transform(ref_cloud[:, :3], R, t)
            #src_cloud_normal = transform(ref_cloud[:, 3:], R)
            #src_cloud = np.concatenate([src_cloud_points, src_cloud_normal],
            #                           axis=-1)
            src_cloud = random_select_points(src_cloud, m=self.npts)
            return src_cloud, ref_cloud, R, t


        elif mode == 'partial':
            #source_cloud = random_select_points(ref_cloud, m=self.npts)
            ref_cloud = random_select_points(ref_cloud, m=self.npts)
            #src_cloud_points = transform(source_cloud[:, :3], R, t)
            #src_cloud_normal = transform(source_cloud[:, 3:], R)
            #src_cloud = np.concatenate([src_cloud_points, src_cloud_normal],
            #                           axis=-1)
            src_cloud = random_select_points(src_cloud, m=self.npts)
            #src_cloud = random_crop(src_cloud, p_keep=0.7)
            return src_cloud, ref_cloud, R, t
        elif mode == 'noise':
            #source_cloud = random_select_points(ref_cloud, m=self.npts)
            ref_cloud = random_select_points(ref_cloud, m=self.npts)
            #src_cloud_points = transform(source_cloud[:, :3], R, t)
            #src_cloud_normal = transform(source_cloud[:, 3:], R)
```

```
            #src_cloud = np.concatenate([src_cloud_points, src_cloud_normal],
            #                           axis=-1)
            src_cloud = random_select_points(src_cloud, m=self.npts)
            return src_cloud, ref_cloud, R, t
        else:
            raise NotImplementedError

    def __getitem__(self, item):
        src_cloud, ref_cloud, R, t = self.compose(mode=self.mode, item=item)
        if self.train or self.mode == 'noise' or self.mode == 'partial':
            ref_cloud[:, :3] = jitter_point_cloud(ref_cloud[:, :3])
            src_cloud[:, :3] = jitter_point_cloud(src_cloud[:, :3])
        if not self.normal:
            ref_cloud, src_cloud = ref_cloud[:, :3], src_cloud[:, :3]
        return ref_cloud, src_cloud, R, t

    def __len__(self):
        return len(self.data)
```