目　录

6.29

- VM 文档全局变量模块

全局变量在模块结果中订阅参数和在全局变量模块订阅目标输出

- 可以看一下别人实现 VM 文档案例时的笔记 visionmaster- CSDN 搜索
- 3DMVS 还剩数据块控制和传输层控制没有看，不知道有用没

✓ 学会了如果进行拼图展现深度图、3D 点云图

使用触发控制，在没有外接编码器的情况下可以使用帧触发的软触发来实现出图

✓ 通读了一遍 VM 文档，没读完，实现了部分案例

6.30

➢ 3DMVS 和 VM 的数据传输关系，VM 上好像没办法操控相机进行取流，那么是需要使用 3DMVS 取到自己需要的图像之后，再打开 VM 进行图像处理吗?

- 应该是可以通过相机参数模块对相机进行调参然后取流

- 尝试了一下全局触发以及 VM 的通讯功能
- 对于 VM 流程图的一些部件和其中的参数还是比较混沌

周报（6.29-6.30）

对相机进行实操

组装相机过程遇到了电源适配器电压不足无法正常启动的情况

练习了 3DMVS 的取流操作以及触发控制

在 VM 平台使用 2D 图做了一些参考案例

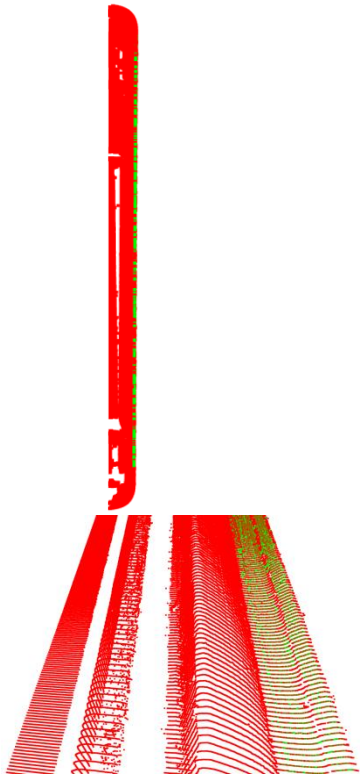
周报（7.3 -7.7）

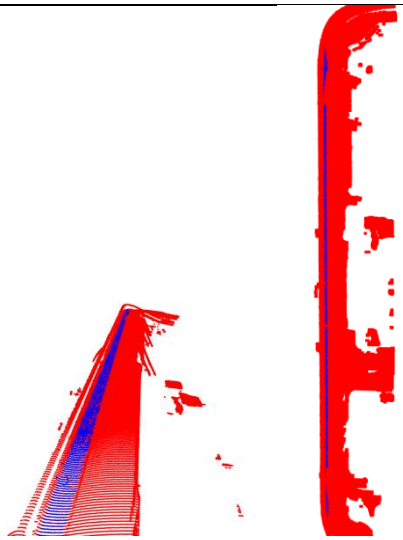
通读了 open3D 的文档，并了解了一些其他的关于点云处理的 python 第三方库

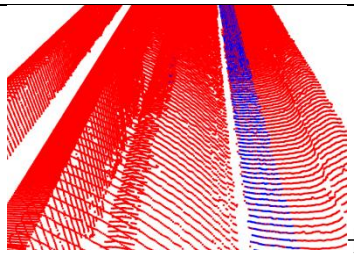
通过手机涂胶工艺轨迹提取，学习到了点云的基本操作，包括点云滤波预处理，点云分割和聚类。

手机涂胶工艺轨迹提取目前效果不太理想，第一就是泛化性太弱，针对性太强，都是针对已有的点云数据进行滤波分割，没有通用性。

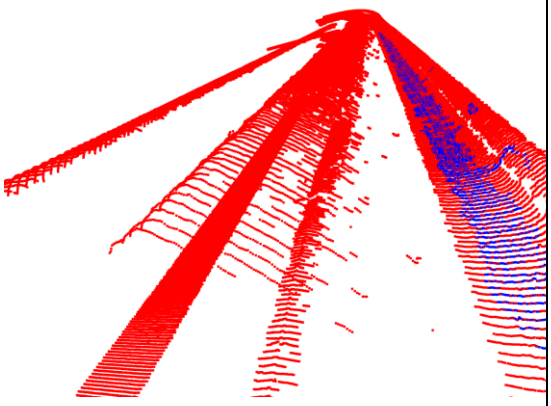
就目前手里的数据而言已有成果就是短边提取较为成功，长边的点云较难提取。







长边偶尔会提取成功



提取长边时，在进行分割前后进行统计->半径滤波----------还是随缘

工作安排（7.10 – 7.14）

*1. 对涂胶工艺轨道算法进行优化 – 尝试新的方法

虽然没有尝试新的方法，但是基本实现了功能

*2. 尝试进行点云拼接

*3. 了解手眼标定算法

*4. 了解机器人轨迹生成为主，重点关注轨迹规划及图形化交互

*5. 进行初步的可行性分析

6. 了解 PCL 和一些其他处理点云的第三方库

7. 了解目前技术的研究现状

8. 出一份初步的可行性报告

7.10

日工作计划:

1. 对涂胶工艺轨道算法进行改进

第一版：平面分割->聚类分割->滤波

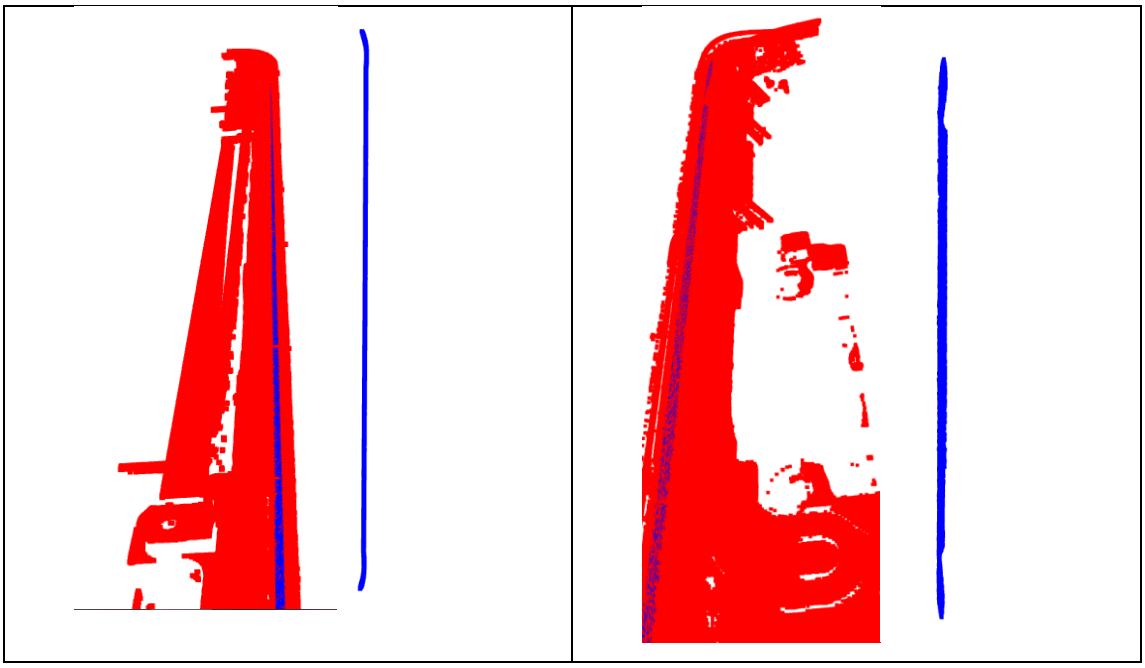
第二版：滤波->平面分割->聚类分割->滤波->(聚类/平面分割)

最终版：聚类分割->平面分割->滤波->聚类分割

最终版缺陷：对于点云残缺较多的点云图无法提取工艺轨道

预期解决方案：首先进行点云缺陷填补

********最终版方案对于左边长边来说不行，左长边(long2)需要在第二次滤波选择第一大簇(解决方案目前没有想好)



2. 细看 open3D 文档（未看完）

在 cvtutorials.com 细看了 open3D 文档，没看完

7.11

1. 了解工程技术线

找一些论文，了解当前技术发展（可以再改进一下轨道提取代码）

7.12

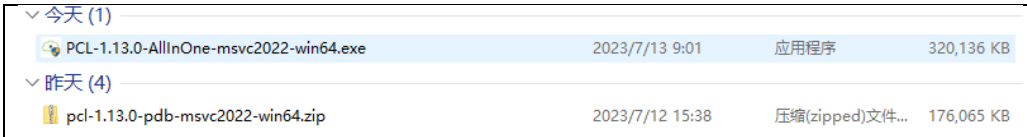
Ubuntu 官方的虚拟机 multi pass 有 bug，后来选择了 virtualbox

7.13

安排：学习 ubuntu，PCL。画出项目流程图并使用 VM 模拟项目流程。博士论文读完。了解其他工具站。查看群宾专利。

1 （环境配置）工作&问题记录

1.1 Windows+vscode+pcl



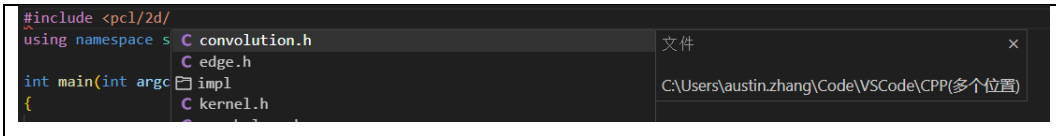
两个都需要安装，pdb 那个文件解压缩到 PCL\bin 里

C:\Users\austin.zhang\DEV\PCL_1_13_0\3rdParty\OpenNI2

这个文件可能会安装到 C:\Program Files

6 个环境变量

Windows 下使用 vscode 安装 pcl 很不友好，目前存在问题：识别不了头文件，再 includePath 中添加路径也没有用 || 感觉 includePath 知识为了头文件不全



路径只是不对，路径指示在当前路径下

```
"includePath": [
    "${workspaceFolder}/**",
    "C:/Users/austin.zhang/DEV/PCL_1_13_0/3rdParty/Boost/include/boost-1_80/**",
    "C:/Users/austin.zhang/DEV/PCL_1_13_0/3rdParty/Eigen/eigen3/**",
    "C:/Users/austin.zhang/DEV/PCL_1_13_0/include/pcl_1_13/pcl",
    "C:/Users/austin.zhang/DEV/PCL_1_13_0/include/pcl_1_13/**",
    "C:/Users/austin.zhang/DEV/PCL_1_13_0/3rdParty/VTK/include/vtk-9.2/**"
],
```

使用**递归，至少头文件路径没问题 -- 可能需要重启 vscode

补充：

Includepath 只是帮助插件找到文件，而报错找不到文件是因为编译找不到，所以要为编译器指定文件路径。（！！！！必要的时候关闭 coderrunner，他的报错信息不完整）

```
"args": [
    "-fdiagnostics-color=always",
    "-g",
    "${file}",
    "-o",
    "${fileDirname}\\${fileBasenameNoExtension}.exe",
    "-I",
    "D:/DEV/PCL_1_13_0/3rdParty/Boost/include/boost-1_80",
    "-I",
    "D:/DEV/PCL_1_13_0/3rdParty/Eigen/eigen3",
    "-I",
    "D:/DEV/PCL_1_13_0/include/pcl-1.13",
    // -I D:/DEV/PCL_1_13_0/3rdParty/Boost/include/boost-1_80 -I
],
"options": {
```

```
'cpp': "cd $dir && g++ -fexec-charset=utf-8 $fileName -I D:/DEV/PCL_1_13_0/3rdParty/Boost/include/boost-1_80 -I D:/DEV/PCL_1_1
```

（coderunner）

这些库是相互依赖的，关闭 coderunner 后可以看到清晰的报错，一个一个添加依赖

```
In file included from D:/DEV/PCL_1_13_0/include/pcl-1.13/pcl/io/pcd_io.h:43:0,
                 from test.cpp:2:
D:/DEV/PCL_1_13_0/include/pcl-1.13/pcl/pcl_macros.h:397:2: error: #error aligned_malloc not supported on your platform
 #error aligned_malloc not supported on your platform
  ^~~~~
D:/DEV/PCL_1_13_0/include/pcl-1.13/pcl/pcl_macros.h:415:2: error: #error aligned_free not supported on your platform
 #error aligned_free not supported on your platform
  ^~~~~
```

这是新的问题，提示平台不支持 aligned_malloc 和 aligned_free

之前 mingw 下载成 win32 版本的了，更换 mingw 版本后 pcl 的头文件总是报错，放弃

```
D:/DEV/PCL_1_13_0/3rdParty/VTK/include/vtk-9.2/vtkCellArray.h:1579:18: warning: 'vtkIdType vtkCellArray::InsertNextCell(vtkIdType, const
vtkIdType*)' redeclared without dllimport attribute after being referenced with dll linkage
 1579 | inline vtkIdType vtkCellArray::InsertNextCell(vtkIdType npts, const vtkIdType* pts)
      |                  ^~~~~~~~~~~~
D:\DEV\MinGW_64\bin/ld.exe: C:\Users\ZhangZB\AppData\Local\Temp\ccw8f144.o:test.cpp:(.text+0xfe): undefined reference to
`__imp__ZN6vtksys18SystemToolsManagerD1Ev'
D:\DEV\MinGW_64\bin/ld.exe: C:\Users\ZhangZB\AppData\Local\Temp\ccw8f144.o:test.cpp:(.text+0x120): undefined reference to
`__imp__ZN20vtkDebugLeaksManagerD1Ev'
D:\DEV\MinGW_64\bin/ld.exe: C:\Users\ZhangZB\AppData\Local\Temp\ccw8f144.o:test.cpp:(.text+0x142): undefined reference to
`__imp__ZN31vtkObjectFactoryRegistryCleanupD1Ev'
D:\DEV\MinGW_64\bin/ld.exe: C:\Users\ZhangZB\AppData\Local\Temp\ccw8f144.o:test.cpp:(.text+0x20e): undefined reference to
`__imp__ZN6vtksys18SystemToolsManagerC1Ev'
D:\DEV\MinGW_64\bin/ld.exe: C:\Users\ZhangZB\AppData\Local\Temp\ccw8f144.o:test.cpp:(.text+0x230): undefined reference to
`__imp__ZN20vtkDebugLeaksManagerC1Ev'
D:\DEV\MinGW_64\bin/ld.exe: C:\Users\ZhangZB\AppData\Local\Temp\ccw8f144.o:test.cpp:(.text+0x252): undefined reference to
`__imp__ZN31vtkObjectFactoryRegistryCleanupC1Ev'
collect2.exe: error: ld returned 1 exit status
```

- 关于 virtualBox 虚拟机启动异常和无法保存设置 ———— 使用管理员权限启动

1.2 Ubuntu+PCL 安装问题

```
sudo apt-get update
sudo apt-get install git build-essential linux-libc-dev
sudo apt-get install cmake cmake-gui
sudo apt-get install libusb-1.0-0-dev libusb-dev libudev-dev
sudo apt-get install mpi-default-dev openmpi-bin openmpi-common
sudo apt-get install libflann1.9 libflann-dev
sudo apt-get install libeigen3-dev
sudo apt-get install libboost-all-dev
sudo apt-get install libqhull* libgtest-dev
sudo apt-get install freeglut3-dev pkg-config
sudo apt-get install libxmu-dev libxi-dev
sudo apt-get install mono-complete
sudo apt-get install libopenni-dev
sudo apt-get install libopenni2-dev
```

批量安装了这些依赖，不知道有用没

Vim xx.sh -> sh xx.sh（就可以批量执行命令）

```
sudo apt install libpcl-dev
```

看了另一个教程，一个命令就可以了，后续具体操作看下面链接

(145 条消息) Ubuntu20.04 Ubuntu18.04 安装 pcl 点云库_长沙有肥鱼的博客-CSDN

博客

Ubuntu 可在终端运行 PCL，windows 未成功

7.14

1. 环境配置

1.1 惨败案例 – windows+vscode+PCL

*针对 windows 环境使用 vscode 开发 pcl 的环境配置问题，应该从 cmake 的配置和编译器在编译过程中寻找头文件路径下手

*先尝试使用 cmake-gui 编译，看是否能通过

（windows + vscode + pcl）太麻烦，windows 使用 vs， ubuntu 使用 vscode

1.2 Ubuntu 安装/远程图形化界面

Ubuntu 安装图形化界面及远程连接（VNC 登录）

进入 root

sudo -s

update 一下

apt-get update

安装 x-winodws

sudo apt-get install x-window-system-core

安装登陆管理器

sudo apt-get install gdm3

安装 ubuntu 桌面

sudo apt-get install ubuntu-desktop

安装 gnome 套件

sudo apt-get install gnome-panel gnome-settings-daemon metacity nautilus gnome-termi

4.然后输入

reboot 重启

————————————————

原文链接：https://blog.csdn.net/wuyihao123/article/details/127479795）

vim ~/.vnc/xstartup

这个配置可以出来文件夹，但是只能出来文件夹

```
export XKL_XMODMAP_DISABLE=1
unset SESSION_MANAGER
unset DBUS_SESSION_BUS_ADDRESS
#!/bin/sh
# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
x-window-manager &
gnome-panel &
gnmoe-settings-daemon &
metacity &
nautilus &
gnome-terminal &
```

VNC 远程图形化界面好像真的只有文件夹而不是整个桌面，遂放弃。

解决方案，windows(mstsc) + ubuntu(xrdp) 需要设置 root 用户和密码，刚开始连接很卡顿，应该在 windows 界面性能里面转换一下网络。

此外，根据下面的文章对 xrdp 进行了优化。

(146 条消息) Xrdp 体验优化 减少/解决画面卡顿_xrdp 卡顿_wuweijie@apache.org 的博客-CSDN 博客

刚开始查的是只能用 root 用户登录，后来莫名其妙的其他用户也可以登录了

2. 论文 - 基于 3D 机器视觉的工业机器人跟踪涂胶系统

1.1 *了解 – 绪论部分相关文献:

1.本文依托于 AI 工业视觉公司“北京阿丘科技有限公司”的生产研发项目

2.哥伦比亚大学 P．ALLEN　　跟踪定位于抓取系统　该系统为 Puma560 机器人增添加了捕捉移动目标的实时视觉

3. 日本 FAUNC 机器人公司研发的“高速双臂工业机器人分拣系统”

4. Murakami．S 等学者研究了基于模糊逻辑控制器的焊缝跟踪控制系统

Murakami S, Takemoto F, Fujimura H, et al. Weld-line tracking control of arc welding robot using fuzzy logic controller[J]. Fuzzy Sets and Systems, 1989, 32(2): 221-237.

5. Jae Seon Kim[15]和 W. P. Gu[16]分别提出了基于线结构光三维测量系统的工业机器人焊接轨迹跟踪的方法

- 采取D-H 法进行涂胶机器人的运动学建模

Gocator -2340 三维激光传感器的 X 方向（激光线方向）轮廓点数达到 1280 个，Y 方向（扫描方向）根据扫描距离的设定可以达到几 K 的长度，CCD 拍摄的帧率能达到 170-5000 帧/秒，非常适用于连续的带状物体的检测。传感器厂商开发了基于浏览器的图形图像用户界面，可以进行参数设置和三维数据实时可视化，并提供了开源 SDK、本地驱动等软件包；传感器本身支持通用的工业通讯协议，方便与其他用户应用软件、第三方图像处理软件和 PLC 进行集成运用。本论文便是基于此款传感器的 SDK 编写了 3D 视觉涂胶系统的图像处理模块程序，用于对采集到的鞋底鞋样表面三维数据进行二次处理，以实现目标三维信息（如涂胶路径点）的提取功能。
本章对 3D 视觉涂胶系统的标定原理进行了理论分析和介绍。标定过程包含 3D 视觉系统标定和涂胶机器人手眼标定两大部分。本章最后编写了基于 Halcon 视觉库的标定程序，设计标定板并进行了标定实验，获取了本文 3D 视觉涂胶系统的一系列标定参数。

- 这篇论文具体技术细节帮助不大，但是里面的参考文献和提到的方法可以一看

小结

不要将目光局限于市面上的涂胶技术，类似的焊接等技术也要多看

在撰写论文的过程中，重点肯定是项目的创新点和技术难点，但是所使用的现有技术

也可以对其进行剖析，例如本项目的3D扫描相机，以及以后会改进的一些算法，在撰写论文的时候可以从其优缺点、性能等方面进行展开。

周报（7.10-7.14）

*1. 对涂胶工艺轨道算法进行优化 – 尝试新的方法

虽然没有尝试新的方法，但是基本实现了功能

*2. 尝试进行点云拼接

*3. 了解手眼标定算法

*4. 了解机器人轨迹生成为主，重点关注轨迹规划及图形化交互

*5. 进行初步的可行性分析

6. 了解 PCL 和一些其他处理点云的第三方库

7. 了解目前技术的研究现状

8. 出一份初步的可行性报告

上周任务基本没有完成，主要是被很多 bug 和环境配置缠住，在使用 ubuntu 的 multi pass 时浪费太多时间，配置 windows+vscode+pcl 也浪费了很多时间，在解决一个新问题的时候设定一个目标时间，如果没有解决就需要换一个解决办法，或者暂停一下。

总结：找了一些开源虚拟机，在上面部署了基于 ubuntu 的 pcl 开发环境，了解了一下 pcl 处理点云的操作。针对市面上一些可以进行 3d 点云处理任务的软件进行了简单调研，类似于 VM 之类的。找到了一些比较贴合项目的论文，目前正在研读和归纳总结。

工作安排（7.17-7.21）

1. 配置 windows 下的 pcl 环境，熟悉一下 c++和 pcl 操作，就拿手机涂胶工艺轨道进行练手
2. 尽快进行可行性分析，思路：完整项目的论文、群宾等公司的专利、相关技术点的论文、市面上相关的软件(VM， Halcon)
3. 三人进行讨论，争取拿出初步的可行性报告

7.17

工作安排：对 VM 和 Halcon 进行调研(使用 VM 做一下手机涂胶轨道提取 – 这玩意只能导入.raw 格式的本地图像，后续考虑使用 Halcon 吧)，根据相关论文画出初步的对于项目理解的流程图。在 Ubuntu 环境下尝试学习 PCL 和 c++。可以考虑配置 windows 环境下的 PCL 环境。学会如何获取海康相机数据。

1.通过 sdk 方式获取海康相机数据

C:\Users\austin.zhang\DEV\3DMVS\Development\Mv3dLpSDK

感觉比较复杂，回头可以跑一下官方给的样例

(146 条消息) C#快速调用海康威视工业相机的 SDK 拍照获取图片_zls365365 的博客-CSDN 博客

可以参考这个例子

2. 环境配置（ubuntu+vscode windows+vs）PCL

2.1 cmake 安装

ubuntu 20.04 安装(升级)cmake - 知乎 (zhihu.com)

(146 条消息) ubuntu 安装 cmake_yuanzhoulvpi 的博客-CSDN 博客

下回要安装新的东西时，记得查看是否有旧版本的。这次安装是 cmake3.25，又是可以 make 成功，有时候显示当前运行版本不是 cmake3.25，运行失败

2.2 vscode 部署 pcl（ubuntu-cmake）

CmakeLists.txt

```
cmake_minimum_required(VERSION 3.25 FATAL_ERROR)
project(pcl_test2)
find_package(PCL 1.10 REQUIRED)
```

```
include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
add_definitions(${PCL_DEFINITIONS})
add_executable(pcl_test2 pcl_test2.cpp)
target_link_libraries(pcl_test2 ${PCL_LIBRARIES})
```

配置了 Cmake 就不用配置其他文件了（包括 includepath）更详细操作如下：

(146 条消息) ubuntu 上 vscode 使用 cmake 编译运行 c++程序_ubuntu vscode c++_SCH0 的博客-CSDN 博客

2.3 windows+vs+PCL

Windows11+VS2022+PCL1.13.0 安装配置记录_WoooChi 的博客-CSDN 博客

属性文件可以重复利用，不用每次都设置。

项目 - 属性 - 调试 - 环境 - 编辑—添加环境如下(每次都要设置)

PATH=$(PCL_ROOT)\bin;$(PCL_ROOT)\3rdParty\FLANN\bin;$(PCL_ROOT)\3rdParty\VTK\bin;$(PCL_ROOT)\3rdParty\Qhull\bin;$(PCL_ROOT)\3rdParty\OpenNI2\Tools;$(PATH)

问题 1. 运行简单生成的点云文件没有问题，读取 pcd 文件会报错

```
#include<iostream>
#include<pcl/io/pcd_io.h>
#include<pcl/io/ply_io.h>
#include<pcl/visualization/cloud_viewer.h>
#include<pcl/io/io.h>
```

Io.h 必须在 cloud_viewer.h 下面，不然 io.h 就会报错

7.18

工作安排：1.研读基于线激光的水轮机机器人测...标定与焊点加工区域特征提取_吉鹏晖，并结合之前关于涂胶的论文，整理出整个项目的大概思路。2.调研 VM、Halcon 和市面上其他商用的视觉算法。3.学习 PCL，整理 PCL 和 open3D 的差异 4.调研海康相机

（将这篇文档转换为 markdown，下载 git，以后直接上传云端）

1. 论文 - 基于线激光的水轮机机器人测...标定与焊点加工区域特征提取

文中有很多参考文献，和国内外相关技术的研究路线

1.1 *手眼标定 --- 相关文章

1. 感觉文章很好，但是没有看懂

(147 条消息) 工业机器人工具坐标系（TCF）标定的六点法原理_工业机器人 tcp 六点法_-贝塔-的博客-CSDN 博客

2.C:\Users\austin.zhang\Documents\Paper\reading

一种结合 TCP 标定的深度相机手眼标定方法

3.超级详细！！

(147 条消息) 机器人手眼标定 Ax=xB（eye to hand 和 eye in hand）及平面九点法标定_yaked19 的博客-CSDN 博客

4.博主的文章比较全面

(147 条消息) 机器人手眼标定原理介绍（含详细推导过程）使用 Tsai-Lenz 算法_鱼香 ROS 的博客-CSDN 博客

1. AX=XB 问题的计算 --- 精度

1.2 *点云处理

点云分割算法主要有以下四大类：基于点云聚类的、基于区域生长的、基于边缘识别的、基于标准模型的。

基于点云聚类的算法中欧式聚类使用较为广泛，其适合在已经分割划分之后对剩下的部分点云进行聚类以便后续的进一步分割。基于边缘识别的方法则适用于在分割刚开始时先利用边缘信息进行预分割。基于标准模型的方法有 Hough 变换和 RANSAC（随机样本一致性），其中 Hough 变换适合提取直线、圆等特征， RANSAC 则多用于平面、圆柱面、球面等。基于区域生长的分割则关注点云中点与点之间的特征信息，通过这样的特征信息将点与点进行合并或者分开。

7.19

工作安排：1.研读基于线激光的水轮机机器人测...标定与焊点加工区域特征提取_吉鹏晖，并结合之前关于涂胶的论文，整理出整个项目的大概思路。2.调研 VM、Halcon 和市面上其他商用的视觉算法。3.学习 PCL，整理 PCL 和 open3D 的差异 4.调研海康相机

（将这篇文档转换为 markdown，下载 git，以后直接上传云端）

1. *点云处理 – 昨天论文

从左至右依次为欧式聚类、区域生长、边缘识别、 RANSAC 算法的效果在点云分割中除了上述分割方法外还有对点云的语义分割（PCSS）， PCSS 相比于 PCS 会在分割后为每个点生成语义信息。 PCSS 的常规的方法为有监督机器学习，目前较为前沿的还有点云的深度学习方法

2. C++ & PCL 学习

2.1 区域生长分割

(149 条消息) pcl 小知识（四）——区域生长分割原理(region growing segmentation)_pcl 分割_刘坤的博客的博客-CSDN 博客

看着效果还可以，应该可以实现提取优化

2.2 提取聚类分类

(149 条消息) pcl 点云聚类后的点云索引提取与输出 pcd 聚类结果_菜是菜人是真帅的博客-CSDN 博客

```
//迭代访问点云索引clusters，直到分割出所有聚类
int max = 0;
pcl::PointCloud<pcl::PointXYZ>::Ptr max_cluster(new pcl::PointCloud<pcl::PointXYZ>);
for (std::vector<pcl::PointIndices>::const_iterator it = clusters.begin(); it != clusters.end(); ++it)
{
    pcl::PointCloud<pcl::PointXYZ>::Ptr cluster(new pcl::PointCloud<pcl::PointXYZ>);

    //创建新的点云数据集cloud_cluster，将所有当前聚类写入到点云数据集中

    for (std::vector<int>::const_iterator pit = it->indices.begin(); pit != it->indices.end(); ++pit)

    cluster->points.push_back(cloud->points[*pit]);
    cluster->width = cluster->points.size();
    cluster->height = 1;
    cluster->is_dense = true;

    //保存聚类结果
    if (cluster->points.size() <= 0)
        break;

    std::cout << "点云" << cluster->points.size() << "有这么多点" << std::endl;
    if (cluster->points.size() > max) {
        max = cluster->points.size();
        max_cluster = cluster;
    }
}
std::stringstream ss;
ss << "long_grow_1"<< ".pcd";
pcl::io::savePCDFile(ss.str(), *max_cluster);
```

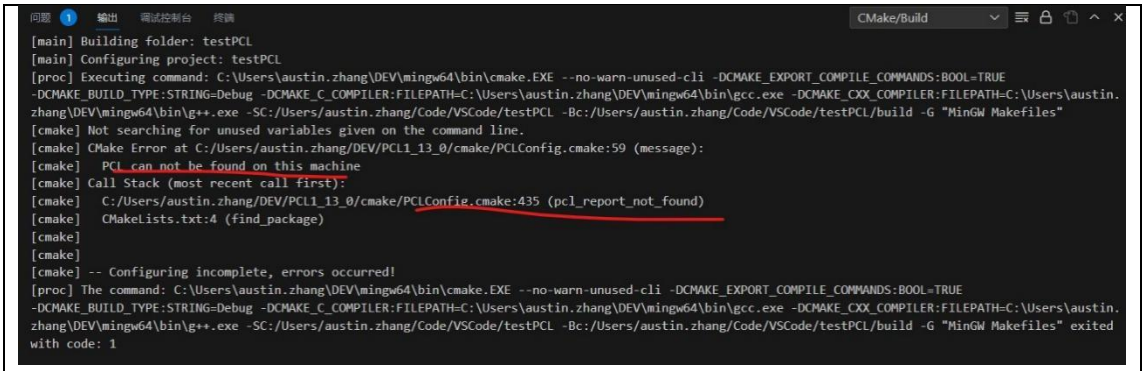
提取最值

2.3 长边提取示例

open3d 聚类 long_utility.pcd -> pcl 区域生长 long_grow0.pcd

```
//聚类对象<点，法线>
pcl::RegionGrowing<pcl::PointXYZ, pcl::Normal> reg;
reg.setMinClusterSize(500);  //最小的聚类的点数
reg.setMaxClusterSize(1000000);  //最大的
reg.setSearchMethod(tree);    //搜索方式
reg.setNumberOfNeighbours(40);    //设置搜索的邻域点的个数
reg.setInputCloud(cloud);         //输入点
//reg.setIndices (indices);
reg.setInputNormals(normals);     //输入的法线
reg.setSmoothnessThreshold(3.0 / 180.0 * M_PI);  //设置平滑度
reg.setCurvatureThreshold(0.5);     //设置曲率的阈值
```

3.windows+vscode+pcl+cmake 问题剖析



```
434   else()
435   |  pcl_report_not_found("${PCL_ROOT}     PCL can not be found on this machine")
436   endif()
437
438   set(PCL_INCLUDE_DIRS "${PCL_CONF_INCLUDE_DIR}")
439
440   #set a suffix for debug libraries
441   set(PCL_DEBUG_SUFFIX "d")
442   set(PCL_RELEASE_SUFFIX "")
443
444   set(PCL_SHARED_LIBS "ON")
445
```

问题 1 输出 调试控制台 终端

```
[proc] Executing command: C:\Users\austin.zhang\DEV\mingw64\bin\cmake.EXE -SC:/Users/austin.zhang/Code/VS
VSCode/testPCL/build -G "MinGW Makefiles"
[proc] The command: C:\Users\austin.zhang\DEV\mingw64\bin\cmake.EXE -SC:/Users/austin.zhang/Code/VSCode/t
testPCL/build -G "MinGW Makefiles" exited with code: 1
[main] Configuring project: testPCL
[proc] Executing command: C:\Users\austin.zhang\DEV\mingw64\bin\cmake.EXE --no-warn-unused-cli -DCMAKE_EX
-DCMAKE_BUILD_TYPE:STRING=Debug -DCMAKE_C_COMPILER:FILEPATH=C:\Users\austin.zhang\DEV\mingw64\bin\gcc.exe
zhang\DEV\mingw64\bin\g++.exe -SC:/Users/austin.zhang/Code/VSCode/testPCL -Bc:/Users/austin.zhang/Code/VS
[cmake] Not searching for unused variables given on the command line.
[cmake] CMake Error at C:/Users/austin.zhang/DEV/PCL1_13_0/cmake/PCLConfig.cmake:59 (message):
[cmake]   C:/Users/austin.zhang/DEV PCL can not be found on this machine
[cmake] Call Stack (most recent call first):
[cmake]   C:/Users/austin.zhang/DEV/PCL1_13_0/cmake/PCLConfig.cmake:435 (pcl_report_not_found)
[cmake]   CMakeLists.txt:7 (find_package)
[cmake]
```

莫名其妙，环境变量和 CmakeLists.txt 都设置了 PCL_ROOT，结果还是错的

4. 海康相机 – 调研+实践

4.1 使用 SDK 拿去海康相机数据

海康有给简单的示例，但是功能不多

4.2 拍出钢板点云图，使用 VM 进行处理，构建基本思路

？钢板横截面拿相机一扫就出来了，之后滤波一下噪声即可。

对项目要做什么还是不清楚

小结：

又浪费时间调试了一下 widows+vscode+pcl，找出了问题在哪，但不知道为什么有问题。论文看了一点，看到论文采用的点云分割方法后就去动手实践，没有完全做出来，但是感觉效果应该比 open3D 好些。重新连接了海康相机，拍出了钢板的点云图，后续使用 VM 进行一些处理，尝试了一下海康自带的 SDK 实例，感觉功能有点少，需要自己写。明天的话最起码把论文基本过一遍，然后找几篇文献，把流程图和可行性报告的首稿做出来。

7.20

工作安排：1.搞清钢板点云处理的任务需求 2.使用 PCL 和 open3D 对点云提取进行优化 3.论文看完并整理思路，使用 chatgpt 整理出可行性报告初版。

待解决问题：1.钢板的详细场景需求 2.获取点云之后，机械臂如何根据提取的点云进行后续工作

*每天安排的任务量力而行，总结的时候要有目的的总结，应该是通过什么学会了什么，还有什么疑问，后续待解决的问题，然后可以说一些注意事项。针对一样意义不大的工作不应该浪费太多时间，比如 windows+pcl+vscode，半个小时如果解决不了就可以先放下等有时间再解决，搞清楚干一件事的目的，配置这个环境是为了学习 pcl 而不是为了配置环境而配置。

1. 钢板项目

1.1 已解决 - 问题 1：项目场景

钢板是竖着放在钢筋上面，然后在钢板上面覆盖一层，并将其与钢板进行焊接，那么在工作时下面与钢筋是怎样连接的，已经焊接好的吗，在后续焊接中还需要考虑下面的情况吗？

首先，这个工作的任务场景是针对钢板上边缘的焊接，那么针对点云提取就好 关键点在于能不能使用 VM 将一套工作直接做出来

1.2 已解决 - 问题 2：获取点云之后的工作如何进行

获取点云轨迹之后如何根据点云轨迹进行工作，不论是涂胶还是焊接。

获取点云之后需要根据点云进行关键点提取，然后传给机器人进行机器人轨迹生成

2. 论文 - 基于线激光的水轮机机器人测...标定与焊点加工区域特征提取

2.1 (章节 3.2.3)线激光点云去重算法分析与实现

2.1.1 FPS 算法

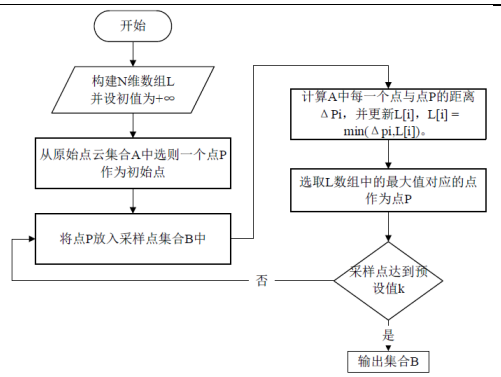


图 3-7 FPS 算法流程

Farthest Point Sampling (FPS)算法核心思想解析 - 知乎 (zhihu.com)

使用了动态规划的思想。

上图解释的很明白，L 数组是 A 中点到点 P 的距离（点 P 是每次新加入的点）

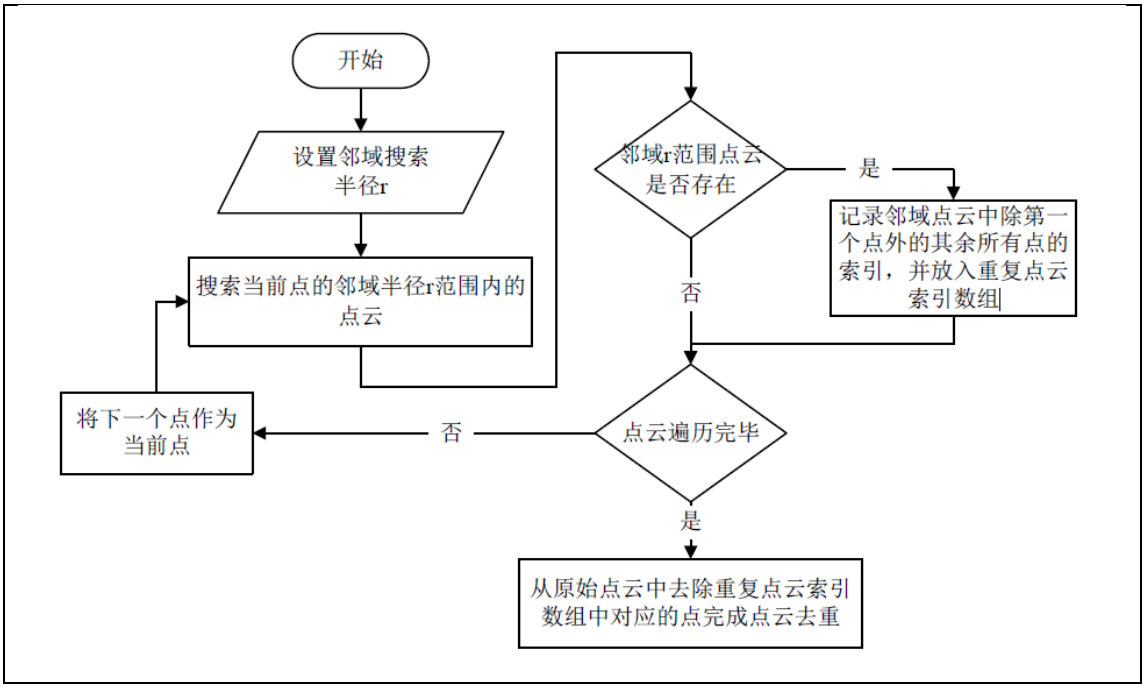
算法的原用和延伸:

点云去重的目的是要去除距离极为接近的点，现有算法中 Farthest Point Sampling(FPS)可以实现这种目的。FPS 算法流程如图 3-7 所示，虽然 FPS 的设计初衷是为了进行降采样，但其“每次从集合 A 中选一个点，使得其到集合 B 里面点的距离最大”的思想可直接用于点云去重。

弊端:

然而 FPS 算法受采样点预设值 k 的影响很大，k 值设置的不好容易导致点云去重不完全或者点云出现缺失。

2.1.2 没看懂 - 使用半径邻域搜索实现去重



2.2 （章节 3.3）线激光点云焊点特征分割算法研究与改进

2.2.1 区域生长分割

该算法的目的是合并在平滑约束条件下足够接近的点。因此，该算法的输出数据结构是由聚类组成的数组，其中每个聚类都是被认为是同一光滑表面的一部分的点的集合。该算法的工作原理（光滑度的计算）是基于两点法线之间的角度比较。

这句话很重要，之前在对手机边框使用聚类进行提取的时候，总是想着把边框直接给聚出来，但是有时候可能反着来更好，把特征明显的区域聚出来，直接扔掉也可以！例如区域生长是找平滑曲面，工艺轨迹附近的平面都比较光滑，且特征明显，可以针对他们进行提取然后丢弃。

搜索到邻域后，这些点先过法线夹角阈值，通过的保留到聚类数据，然后再从这些通过法线夹角阈值的点中，检查是否通过曲率阈值，通过的加入种子点序列。

(149 条消息) 【C++】pcl 中的 Region Growing（区域生长）算法_pcl::regiongrowing_Zhang_Chen_的博客-CSDN 博客

小结：

文章看完，感觉对于整个项目的认知没有变得更清晰，各部分功能实现还是有些割裂。

在本文涉及的实际场景中，由于线激光测量设备的实际扫描范围有限，当需要获取一个大范围区域的表面点云时，应将多次扫描的结果汇总。在多次扫描时必将出现部分区域

进行重复扫描的情况，这就导致最后获得的点云有重复点出现。

3. 改进 - 手机边框点云提取

改变了策略，2.2.1 部分有相关思想，目前使用 open3d 进行滤波处理效果不错，但是分割提取部分应该还需要使用 PCL 来做。

Open3d 滤波思想，对原始点云进行聚类 -> 双重滤波 -> 聚类 -> 双重滤波，好处就是减少平面切割带来的不确定性，提高了算法的鲁棒性，同时使得代替去特征区域更加明显。（就是不知道为什么双重滤波部分很慢）

4. 已解决 - windows+vscode+PCL

4.1 彻底解决方案

更改 vscode 中 kit 位 vs Release 即可！！！！！！！

4.2 未彻底解决 - MinGW 解决方案

```
file(TO_CMAKE_PATH "${PCL_DIR}" PCL_DIR)
if(WIN32 AND NOT MINGW)
# PCLConfig.cmake is installed to PCL_ROOT/cmake
  get_filename_component(PCL_ROOT "${PCL_DIR}" PATH)
  if(EXISTS "${PCL_ROOT}/3rdParty")
    set(PCL_ALL_IN_ONE_INSTALLER ON)
  endif()
else()
# 断点1 目录位置
  # pcl_report_not_found("走了")
# PCLConfig.cmake is installed to PCL_ROOT/share/pcl-x.y
  get_filename_component(PCL_ROOT "${CMAKE_CURRENT_LIST_DIR}/../.." ABSOLUTE)
  # if(EXISTS "${PCL_ROOT}/3rdParty")
  #   set(PCL_ALL_IN_ONE_INSTALLER ON)
  # pcl_report_not_found("${PCL_DIR}\n ${PCL_ROOT}\n  PCL can not be found on this machine")
  endif()
endif()
```

```
file(TO_CMAKE_PATH "${PCL_DIR}" PCL_DIR)
if(WIN32 AND NOT MINGW)
# PCLConfig.cmake is installed to PCL_ROOT/cmake
  get_filename_component(PCL_ROOT "${PCL_DIR}" PATH)
  if(EXISTS "${PCL_ROOT}/3rdParty")
    set(PCL_ALL_IN_ONE_INSTALLER ON)
  endif()
else()
# 断点1 目录位置
  # pcl_report_not_found("走了")
# PCLConfig.cmake is installed to PCL_ROOT/share/pcl-x.y
  get_filename_component(PCL_ROOT "${CMAKE_CURRENT_LIST_DIR}/.." ABSOLUTE)
  if(EXISTS "${PCL_ROOT}/3rdParty")
    set(PCL_ALL_IN_ONE_INSTALLER ON)
  # pcl_report_not_found("${PCL_DIR}\n ${PCL_ROOT}\n  PCL can not be found on this machine")
  endif()
endif()
```

```
[main] Configuring project: Cpp
[proc] Executing command: D:\DEV\CMake\bin\cmake.EXE --no-warn-unused-cli -DCMAKE_
-DCMAKE_C_COMPILER:FILEPATH=D:\DEV\MinGW_64\bin\gcc.exe -DCMAKE_CXX_COMPILER:FILEP
"MinGW Makefiles"
[cmake] Not searching for unused variables given on the command line.
[cmake] CMake Error at D:/DEV/PCL_1_13_0/cmake/PCLConfig.cmake:59 (message):
[cmake]   D:/DEV/PCL_1_13_0/cmake
[cmake]
[cmake]    D:/DEV/PCL_1_13_0
[cmake]     PCL can not be found on this machine
[cmake] Call Stack (most recent call first):
[cmake]   D:/DEV/PCL_1_13_0/cmake/PCLConfig.cmake:420 (pcl_report_not_found)
[cmake]   CMakeLists.txt:5 (find_package)
[cmake]
[cmake]
[cmake] -- Configuring incomplete, errors occurred!
[proc] The command: D:\DEV\CMake\bin\cmake.EXE --no-warn-unused-cli -DCMAKE_EXPORT
-DCMAKE_C_COMPILER:FILEPATH=D:\DEV\MinGW_64\bin\gcc.exe -DCMAKE_CXX_COMPILER:FILEP
"MinGW Makefiles" exited with code: 1
```

修改第二条分支可以找到正确路径，但还是报错。报错信息如下：

```
-DCMAKE_C_COMPILER:FILEPATH=D:\DEV\MinGW_64\bin\gcc.exe -DCMAKE_CXX_COMPILER:FILEPATH=D:\DEV\MinGW_64\bin\g++.exe -SD:/Projects/Vscode/Cpp
-Bd:/Projects/Vscode/Cpp/build -G "MinGW Makefiles"
[cmake] Not searching for unused variables given on the command line.
[cmake] CMake Warning (dev) at D:/DEV/PCL_1_13_0/cmake/PCLConfig.cmake:143 (find_package):
[cmake]   Policy CMP0144 is not set: find_package uses upper-case <PACKAGENAME>_ROOT
[cmake]   variables.  Run "cmake --help-policy CMP0144" for policy details.  Use the
[cmake]   cmake_policy command to set the policy and suppress this warning.
[cmake]
[cmake]   CMake variable EIGEN_ROOT is set to:
[cmake]
[cmake]     D:/DEV/PCL_1_13_0/3rdParty/Eigen
[cmake]
[cmake]   For compatibility, find_package is ignoring the variable, but code in a
[cmake]   .cmake module might still use it.
[cmake] Call Stack (most recent call first):
[cmake]   D:/DEV/PCL_1_13_0/cmake/PCLConfig.cmake:296 (find_eigen)
[cmake]   D:/DEV/PCL_1_13_0/cmake/PCLConfig.cmake:547 (find_external_library)
[cmake]   CMakeLists.txt:5 (find_package)
[cmake] This warning is for project developers.  Use -Wno-dev to suppress it.
[cmake]
[cmake] -- Checking for module 'eigen3'
[cmake] --
[cmake] -- Found Eigen: D:/DEV/PCL_1_13_0/3rdParty/Eigen/eigen3 (Required is at least version "3.3")
[cmake] -- Eigen found (include: D:/DEV/PCL_1_13_0/3rdParty/Eigen/eigen3, version: 3.4.0)
[cmake] CMake Warning (dev) at D:/DEV/PCL_1_13_0/cmake/PCLConfig.cmake:125 (find_package):
[cmake]   Policy CMP0144 is not set: find_package uses upper-case <PACKAGENAME>_ROOT
[cmake]   variables.  Run "cmake --help-policy CMP0144" for policy details.  Use the
[cmake]   cmake_policy command to set the policy and suppress this warning.
[cmake]
[cmake]   CMake variable BOOST_ROOT is set to:
[cmake]
[cmake]     D:/DEV/PCL_1_13_0/3rdParty/Boost
[cmake]
[cmake]   For compatibility, find_package is ignoring the variable, but code in a
[cmake]   .cmake module might still use it.
[cmake] Call Stack (most recent call first):
[cmake]   D:/DEV/PCL_1_13_0/cmake/PCLConfig.cmake:294 (find_boost)
[cmake]   D:/DEV/PCL_1_13_0/cmake/PCLConfig.cmake:547 (find_external_library)
[cmake]   CMakeLists.txt:5 (find_package)
```

这里的 win32 指的不是系统，而是编译器版本 这里是错误的，使用 64 位的 kit，依然走的是第一条分支。

5. 关于 BUG - 小结

不论是配置环境还是平时遇到 BUG，需要仔细查看日志以及报错信息去排查错误，不要急，慢慢排查，不要一直在一条死胡同走!

7.21

工作安排：1.找相关案例和商用软件写可行性报告 2.使用 VM 对钢板点云进行处理并研究海康相机性能 3.对点云提取进行优化 – 顺手把周报写了

1. Halcon

1.1 安装 – 并在 vs 调用 halcon 的包

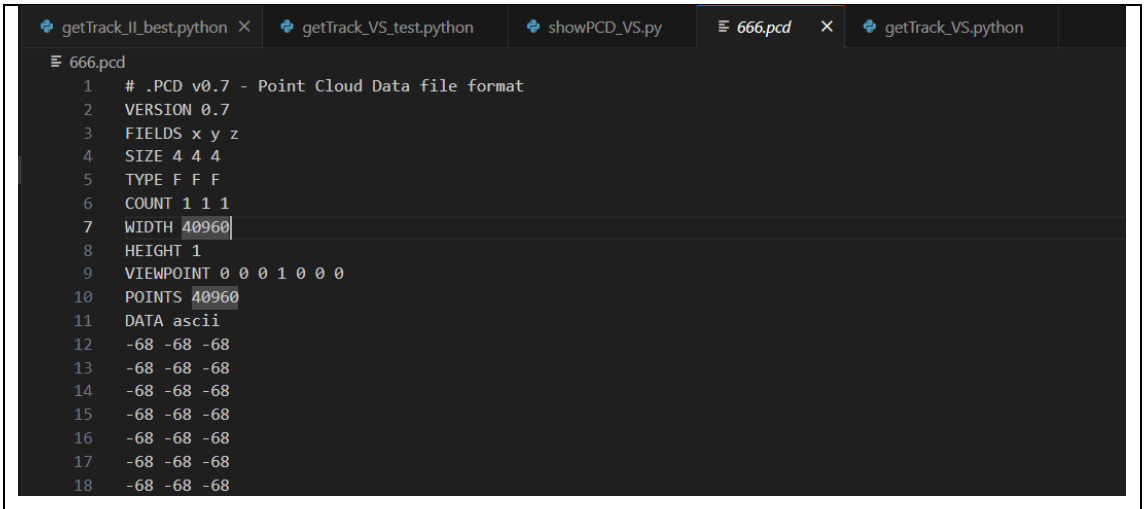
(149 条消息) halcon23.05 下载安装，并在 qt creator 和 vs2022 使用 halcon 的包_稷滶的博客-CSDN 博客

2. pcl 学习路线及资料

PCL(Point Cloud Library)学习指南&资料推荐（2023 版） - 知乎 (zhihu.com) Github 已 star
01-点云及其可视化 - 黑马机器人	PCL-3D 点云 (czxy.com)
pcl- CSDN 搜索

3. 海康相机 – 数据信息

3.1 点云数据 – 深\亮度图



20230721162345.bmp	2023/7/21 16:26	BMP 文件	2,770
20230721162349.raw	2023/7/21 16:23	RAW 文件	2,741

周报（7.17-7.21）

1.了解了如何通过 sdk 直接拿取海康相机数据 2.读完了水轮机的论文，学习了关于整个项目的构建思路 3.了解 PCL 大概的技术功能，并开始细致的学习 PCL 和 C++语法

*读论文要有侧重点，工作时间应该把精力集中在重要的地方

工作安排（7.24-7.28）

1.可行性报告 -> 市场：市面上类似的项目 -> 技术：探索商用视觉算子 -> 技术：本项目要使用的技术。

2.钢板项目，使用 VM 在嘈杂环境下也能把钢板提出来

3.详细学习 C++和 PCL

4.看论文

7.24

1.针对上周调研的内容和以前看的论文写出可行性报告初版

2.使用 VM/Halcon 对钢板点云进行提取

3.学习 C++和 PCL

1. 可行性报告 – 论文

1.1 工业机器人涂胶路径规划与仿真研究

工业机器人技术在智能制造业中的应用越来越广泛，在涂胶领域的应用，可以有效解决人工涂胶效率低、涂胶质量不高、涂胶过程有害气体对人体伤害等问题。通过与传统的工业机器人现场示教点编程生成轨迹路径的方法相比较，采用虚拟仿真软件中“自动路径”功能优化后生成的路径轨迹与几何体曲面轮廓贴合度更好，涂胶厚度更加均匀，涂胶质量和效率更高。

胶接作为一种重要的连接方式，广泛应用于产品的制造过程中。目前，国内仍然有大部分企业采用人工涂胶的方法，手工涂胶的质量受个人熟练程度限制，容易出现涂胶不均匀、不连续和浪费等现象，并且大部分胶易挥发出有毒气体，对人体产生伤害。采用现场示教点位的编程方法可以很好地完成单一平直的涂胶路径轨迹规划，但对于具有复杂曲线的涂胶路径此种方法难以满足要求，会出现示教点位过多、编程工作量大、生产效率低下、涂胶质量差的情况。

1.2 基于视觉的机器人自动化涂胶质量检测技术研究

摘要及绪论部分有很多可以借鉴

7.30

1 使用 PCL 进行点云提取

https://blog.csdn.net/luolaihua2018/article/details/120184539
感觉可以实现，但是 BUG 太多
https://blog.csdn.net/qq_25105061/article/details/119614919

2. Open3D 安装

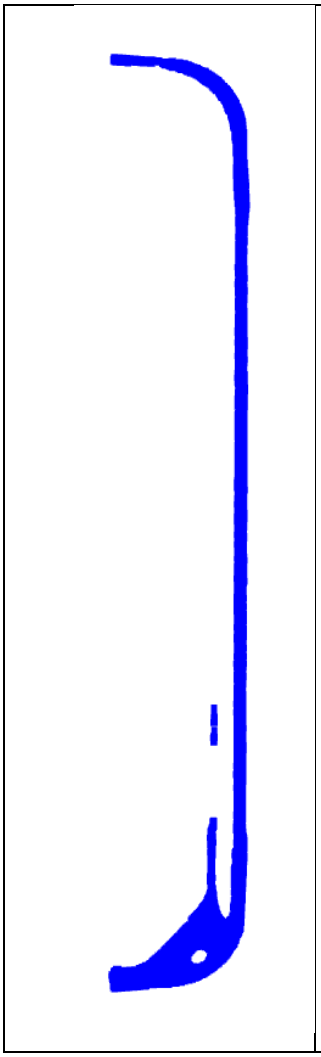
conda install numpy #安装 matplotlib
conda install matplotlib #安装 matplotlib
conda install -c open3d-admin open3d #安装 Open3D

3. 手机胶道改进

```python
filename = "long_right"
isShort = True if 'short' in filename else False

pcd = readOrWrite(filename, "read")
# pcd = removeNoise_statistical(pcd, voxel_size=0.1)
_, pcd = segmentPCD(pcd, distance_threshold=0.09)
pcd = removeNoise_radius(pcd, num_points=15, radius=0.5)
pcd, _ = utility(pcd, eps=0.25)
pcd = removeNoise_statistical(pcd, voxel_size=0.1)
pcd = removeNoise_radius(pcd, num_points=30, radius=0.5)
# pcd = removeNoise_statistical(pcd, voxel_size=0.1)
pcd, _ = utility(pcd, eps=0.8, isDouble=True)
pcd.paint_uniform_color([0,0,1])
pcd = removeNoise_radius(pcd, num_points=30, radius=0.5)

showPointCloud([pcd])
```

```python
filename = "long_left"
isShort = True if 'short' in filename else False

pcd = readOrWrite(filename, "read")
# pcd = removeNoise_statistical(pcd, voxel_size=0.1)
_, pcd = segmentPCD(pcd, distance_threshold=0.09)
pcd = removeNoise_radius(pcd, num_points=15, radius=0.5)
pcd, _ = utility(pcd, eps=0.25)
# pcd = removeNoise_statistical(pcd, voxel_size=0.1)
pcd = removeNoise_radius(pcd, num_points=30, radius=0.5)
# pcd = removeNoise_statistical(pcd, voxel_size=0.1)
pcd, _ = utility(pcd, eps=0.8, isDouble=True)
pcd = removeNoise_radius(pcd, num_points=30, radius=0.5)
pcd.paint_uniform_color([0,0,1])
pcd = removeNoise_radius(pcd, num_points=30, radius=0.5)

showPointCloud([pcd])
```

```python
if __name__ == '__main__':
    filename = "short_left"
    isShort = True if 'short' in filename else False

    pcd = readOrWrite(filename, "read")
    # pcd = removeNoise_statistical(pcd, voxel_size=0.1)
    _, pcd = segmentPCD(pcd, distance_threshold=0.07)
    pcd = removeNoise_radius(pcd, num_points=15, radius=0
    pcd, _ = utility(pcd, eps=0.25)
    pcd = removeNoise_statistical(pcd, voxel_size=0.05)
    pcd = removeNoise_radius(pcd, num_points=30, radius=0
    pcd, _ = utility(pcd, eps=0.6, isDouble=False)
    pcd = removeNoise_radius(pcd, num_points=30, radius=0
    pcd.paint_uniform_color([0,0,1])
    showPointCloud([pcd])
```



如上

7.31

1.Vscode+Docker+PCL

安装 Docker 和配置 VScode 连接 - 知乎 (zhihu.com)

8.1

1.区域生长

(149 条消息) pcl 区域生长算法（一）_长沙有肥鱼的博客-CSDN 博客

(149 条消息) pcl 小知识（四）——区域生长分割原理(region growing segmentation)_pcl 分割_刘坤的博客的博客-CSDN 博客

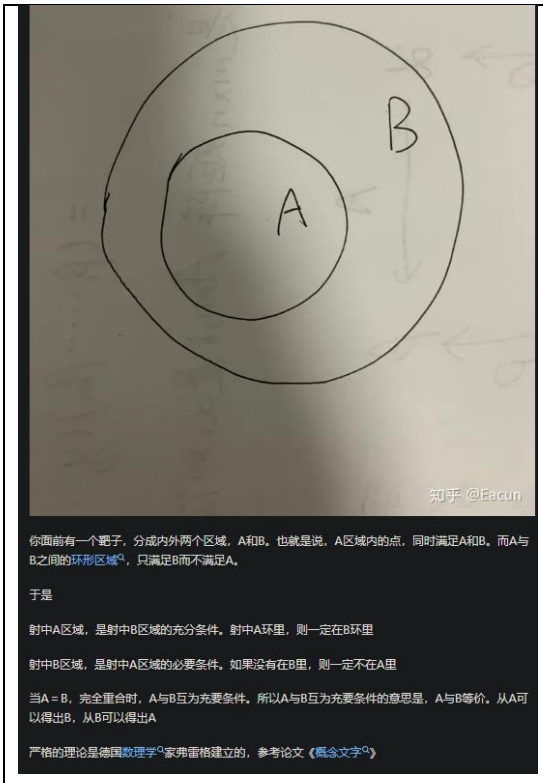
2.计算相机参数 excel 表格

=VLOOKUP(C3,$P:$R,2,0)　根据 C3 的数据获取 P-R 的第二列，C3 数据必须在 P-R 的第一列（0 表示精准查找，1 表示模糊查找）

将 excel 表格保存为 xlsm 即可保存其中的宏或 VB

8.2|8.3

1.充分必要条件



你面前有一个靶子，分成内外两个区域，A和B。也就是说，A区域内的点，同时满足A和B。而A与B之间的环形区域[Q]，只满足B而不满足A。

于是

射中A区域，是射中B区域的充分条件。射中A环里，则一定在B环里

射中B区域，是射中A区域的必要条件。如果没有在B里，则一定不在A里

当A = B，完全重合时，A与B互为充要条件。所以A与B互为充要条件的意思是，A与B等价。从A可以得出B，从B可以得出A

严格的理论是德国数理学[Q]家弗雷格建立的，参考论文《概念文字[Q]》

2. 论文 - 基于线激光的水轮机机器人测...标定与焊点加工区域特征提取

2.1 手眼标定

#手眼标定 (qq.com)

8.14

1.论文-基于标准圆柱的线激光轮廓扫描机器人手眼标定方法

1.1 知识点

1.1.1 罚函数

罚函数的基本思想是，借助罚函数把约束问题转化为无约束问题，进而用无约束最优

化方法求解。

罚函数法 - 知乎 (zhihu.com)

1.1.2 齐次变换矩阵

$$T=\begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}=\begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ r_{31} & r_{32} & r_{33} & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

齐次变换矩阵，R 是旋转矩阵（正交矩阵），p 是平移向量， [0 0 0 1]是为了方便运算

Ps:

$$\begin{pmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix}=\begin{pmatrix} p_1+t_1 \\ p_2+t_2 \\ p_3+t_3 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} I & T \\ O & 1 \end{pmatrix}\begin{pmatrix} P \\ 1 \end{pmatrix}=\begin{pmatrix} IP+T \\ 1 \end{pmatrix}$$

旋转矩阵是正交矩阵的一种，因为旋转矩阵保持向量的长度不变，同时保持向量之间的夹角不变。这意味着，当我们将一个向量乘以一个旋转矩阵时，它不会改变向量的长度，也不会改变向量与其他向量之间的夹角。正交矩阵的定义是满足$Q^TQ=I$的矩阵，其中Q^T是Q的转置矩阵，I是单位矩阵。由于旋转矩阵保持向量的长度和夹角不变，因此它们也满足这个定义，因此旋转矩阵也是正交矩阵。

1.1.3 PSO 粒子群优化算法

粒子群优化算法(Particle Swarm Optimization, PSO)的详细解读 - 知乎 (zhihu.com)

1.1.4 ransac 拟合椭圆

```
import numpy as np
from sklearn.linear_model import LinearRegression, RANSACRegressor
from sklearn.metrics import mean_squared_error

# 生成随机点云
n_samples = 100
x = np.random.uniform(-10, 10, n_samples)
y = np.random.uniform(-5, 5, n_samples)
noise = np.random.normal(0, 1, n_samples)
x += noise
y += noise

# 将点云转换为二维数组
```

```
data = np.column_stack((x, y))

# 定义椭圆模型
def ellipse_model(x, a, b, h, k):
    return ((x[:,0]-h)/a)**2 + ((x[:,1]-k)/b)**2

# 定义 RANSAC 回归器
ransac = RANSACRegressor(base_estimator=LinearRegression(),
                         min_samples=10, residual_threshold=1.0,
                         random_state=0)

# 运行 RANSAC 算法
ransac.fit(data, ellipse_model)

# 计算内点数目和外点数目
inliers = ransac.inlier_mask_
outliers = np.logical_not(inliers)

# 重新拟合椭圆
a, b, h, k = np.linalg.lstsq(data[inliers],
ellipse_model(data[inliers], *ransac.estimator_.coef_), rcond=None)[0]

# 输出拟合结果
print('a =', a)
print('b =', b)
print('h =', h)
print('k =', k)
```

可以参考一下，chatgpt 给的代码，可能不对

1.2 论文笔记

1.2.1 精度问题

论文中使用 RANSAC 对线激光与圆柱相交的点云进行椭圆拟合。使用椭圆估计中心点可能会出现误差。

> 标定算法基于圆柱侧面的约束，即，使所有激光轮廓数据到圆柱侧面的距离之和最小。为方便计算，利用椭圆轮廓估计中心点到圆柱中轴线的距离表示椭圆轮廓数据到圆柱侧面的距离。根据椭圆轮

1.3 论文待解决问题

最优化问题求解部分没有仔细看

2. 论文 - 线激光器的手眼标定方法

2.1 四元数 – 含未解决问题

四元数——基本概念 - 知乎 (zhihu.com) – 这是高阶篇，以后可以看看 不理解 $$X=\begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) & t_1 \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) & t_2 \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

3. 手眼标定论文小结

相同手眼标定方法，其结果精度会存在较大差异，不排除论文数据有问题，从目前两篇文章来看还没有办法得知他们出现精度差别的地方在哪里。

**还有重要的一点，在刚开始接触一项新的技术的时候可以适当的看一些学位论文（综述），之后应该看相关技术的小论文，比较深入了解这项技术之后可以再找一些相关的大论文来看。

8.18

1.wsl & Docker

win10 离线安装 WSL2 Ubuntu20.04 系统 - 简书 (jianshu.com) docker 桌面版报错 error during connect: This error may indicate that the docker daemon is not running.:_程序工厂的博客-CSDN 博客

工作安排(8.21-8.25)

论文，点云代码（提取，特征匹配，特征点提取 - 可以补充一些代码）

论文看完之后要总结，尝试性的看一些外文文献。月度汇报可以再更改总结一下。

8.21

1.最小二乘法

最小二乘法（least sqaure method） - 知乎 (zhihu.com)

8.22

1.手眼标定相关知识

1.1 欧拉角-四元数-旋转矩阵

欧拉角和旋转矩阵之间的转换 - 知乎 (zhihu.com) – 有欧拉角转旋转矩阵的代码

旋转矩阵（Rotation Matrix)的推导 - 知乎 (zhihu.com)

8.23

1. 点云 – 网络

(36 封私信 / 80 条消息) 点云数据可以用来干什么？ - 知乎 (zhihu.com)

8.28

1. 手眼标定相关博客

关于手眼标定的误差计算_手眼标定误差计算_boss-dog 的博客-CSDN 博客

三维旋转：欧拉角、四元数、旋转矩阵、轴角之间的转换 - 知乎 (zhihu.com)

（一）关于手眼标定理论相关的笔记_boss-dog 的博客-CSDN 博客

（二）2D 视觉机器人的手眼标定流程记录_2d 手眼标定动作_boss-dog 的博客-CSDN 博客

（三）手眼标定结果的应用_手眼标定的结果_boss-dog 的博客-CSDN 博客

SLAM 经典文献之：Hand-Eye Calibration（手眼标定） - 知乎 (zhihu.com)

OpenCV 手眼标定（calibrateHandeye()）_hellohake 的博客-CSDN 博客

8.30

1. vs 问题 – 无法 Debug

按照 7.17-2.3 重新配置即可

2. vs 问题 – 使用 pcl 智能指针报错 – 错误代码-1073740940

PCL 求助（一）-- 指针释放_pcl 中 ptr 释放问题_看到我请叫我学 C++的博客-CSDN 博客

VS 可以通过项目属性->C/C++->代码生成->启动增强指令集->选择 AVX 来解决，我用的

是 PCL1.12.1 版本

8.31

1. PointNet – 包含论文解读

[最全 PointNet 和 PointNet++要点梳理总结_X_Imagine 的博客-CSDN 博客]

9.1

1.关于点云提取

不要再随便调参数了，仔细看看法向量计算原理，边界点计算原理。

PCL 智能指针：直接将指针传入函数，函数中使用指针就会对指针进行修改，不需要返回指针。经过更改之后，代码运行速度搜搜快，也不会出现报错问题。

2. 提取成功

终于提取成功了，按照之前的想法，循环即便流程。

但是程序还有待改进，短边滤的有点窄，且右边有一点被截断了

但是这个莫名其妙，这个目标点云(短边)在代码中被认定为边界点 - 参考长边法线角度阈值

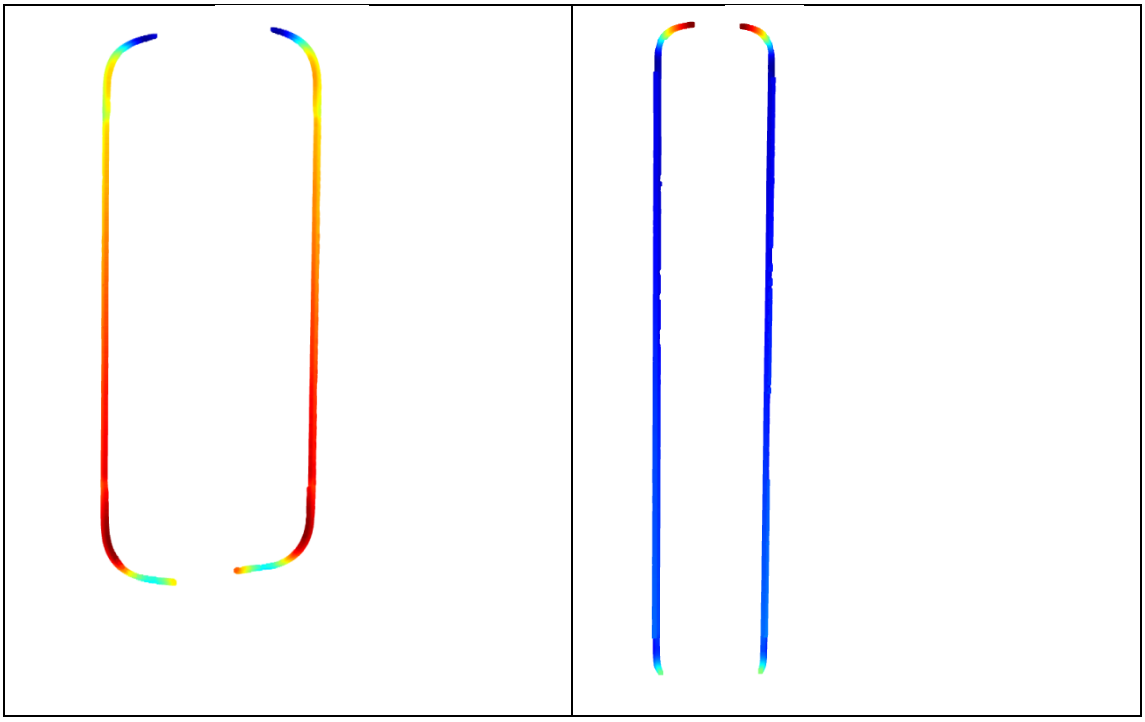
```
for (int i = 0; i < 4; i++) {
    printf("第%d次去除边缘\n", i);
    main_function(cloud, cloud_filted);
    *cloud = *cloud_filted;
    cloud_filted->clear();
}
```

```
void main_function(pcl::PointCloud<pcl::PointXYZ>::Ptr cloud, pcl::PointCloud<pcl::PointXYZ>::Ptr cloud_filted) {

    pcl::PointCloud<pcl::PointXYZ>::Ptr new_cloud(new pcl::PointCloud<pcl::PointXYZ>);

    pcl::PointCloud<pcl::Normal>::Ptr normals(new pcl::PointCloud<pcl::Normal>);
    pcl::PointCloud<pcl::Boundary>::Ptr boundaries(new pcl::PointCloud<pcl::Boundary>);

    std::cout << "PointCloud before filtering has: " << cloud->points.size() << " data points." << std::endl;

    // 统计滤波
    //showPCD(cloud);
    filter_Statistical(cloud, cloud_filted, FALSE);
    // 计算点云法向量 - 参数：周围点数(0)||搜索半径(1)
    computeNormal(cloud_filted, normals, TRUE, 0, 100);
    //  根据法向量计算边界点 - 参数：周围点数（0）||搜索半径(1)and角度阈值（PI/0.7）
    computeEdge(cloud_filted, normals, boundaries, TRUE, TRUE, 0, 50, M_PI * 0.5);
    // 莫名发现现有参数的判定的边界点是提取目标 - 先凑活
    removeEdge(cloud_filted, boundaries, new_cloud, FALSE);
    std::cout << "PointCloud before filtering has: " << new_cloud->points.size() << " data points." << std::endl; //
    // 统计滤波
    filter_Statistical(new_cloud, cloud_filted, FALSE);
    //// 下采样
    //cloud = filter_Voxle(cloud, TRUE);
    //showPCD(cloud);
    std::cout << "PointCloud before filtering has: " << cloud_filted->points.size() << " data points." << std::endl;
}
```

```
int main() {
    pcl::PointCloud<pcl::PointXYZ>::Ptr cloud(new pcl::PointCloud<pcl::PointXYZ>);
    pcl::PointCloud<pcl::PointXYZ>::Ptr cloud_filted(new pcl::PointCloud<pcl::PointXYZ>);
    pcl::io::loadPCDFile("D:/Projects/Vscode/PythonTest/long1.pcd", *cloud);

    filter_Statistical(cloud, cloud_filted, FALSE);
    for (int i = 0; i < 3; i++) {
        printf("第%d次去除边缘\n", i+1);
        main_function(cloud, cloud_filted, i);
        /**cloud = *cloud_filted;
        cloud_filted->clear();*/
    }
    *cloud = *cloud_filted;
    cloud_filted->clear();
    filter_Statistical(cloud, cloud_filted, FALSE);
    Cluster temp;
    // 聚类间最大间距
    temp.ClusterAndShow(cloud_filted, 0.11);

    return 0;
}
```

```
void main_function(pcl::PointCloud<pcl::PointXYZ>::Ptr cloud, pcl::PointCloud<pcl::PointXYZ>::Ptr cloud_filted, int count) {

    pcl::PointCloud<pcl::PointXYZ>::Ptr new_cloud(new pcl::PointCloud<pcl::PointXYZ>);

    pcl::PointCloud<pcl::Normal>::Ptr normals(new pcl::PointCloud<pcl::Normal>);
    pcl::PointCloud<pcl::Boundary>::Ptr boundaries(new pcl::PointCloud<pcl::Boundary>);

    std::cout << "PointCloud before filtering has: " << cloud->points.size() << " data points." << std::endl;

    // 统计滤波
    //showPCD(cloud);
    /*if (count < 2) {
        filter_Statistical(cloud, cloud_filted, FALSE);
    } else {
        *cloud_filted = *cloud;
    }*/
    // 计算点云法向量 - 参数：周围点数(0)||搜索半径(1)
    computeNormal(cloud_filted, normals, TRUE, 0, 100);
    //  根据法向量计算边界点 - 参数：周围点数（0）||搜索半径(1)and角度阈值（PI/0.7）
    computeEdge(cloud_filted, normals, boundaries, TRUE, FALSE, 0, 100, M_PI * 0.6);
    // 莫名发现现有参数的判定的边界点是提取目标 - 先凑活
    removeEdge(cloud_filted, boundaries, new_cloud, FALSE);
    std::cout << "PointCloud before filtering has: " << cloud_filted->points.size() << " data points." << std::endl; //
    // 统计滤波
    if (count < 2)
        filter_Statistical(new_cloud, cloud_filted, FALSE);
    //// 下采样
    //cloud = filter_Voxle(cloud, TRUE);
    //showPCD(cloud);
    std::cout << "PointCloud before filtering has: " << cloud_filted->points.size() << " data points." << std::endl;
}
```

3. 点云提取截图



9.15

1. 手眼标定相关博客

机械臂手眼标定原理及代码_右乘联体左乘基_绿竹巷人的博客-CSDN 博客

10.8

1.失败 - 搭建 socks5 代理服务器

服务器搭建 Socks5 代理详细教程-帮助文档-蓝米云 (lanmicloud.com) 搭建不了啊

2. 突然无法使用 conda 命令

在命令行无法使用 conda 命令，但是 conda --no-plugins 可以，这个命令表示的是禁用所有插件，然后就可以使用了。之后使用 conda --no-plugins update conda 更新 conda 即可，之后 conda 命令就可以用了。

10.10

1.关于访问某些网页很慢的问题

1. 打开网页控制台，查看哪些域名地址访问过慢，使用 ip 测速工具找到一个访问比较快的

站点，然后更改 C:\Windows\System32\drivers\etc\hosts 文件

```
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Windows\system32> cd C:\Windows\System32\drivers\etc
PS C:\Windows\System32\drivers\etc> notepad hosts
```

10.18

求解手眼矩阵(TSAI & PSO)

结论，现有数据差别不大。其次 c++速度比 python 快的不止一个量级。

x,y,z 误差的平均：TSAI 在 0.24 多一点，PSO 也就在 0.23 多一点。

TSAI

平均值	0.28199778570207157	0.20096888177634056	0.2664016288584333	1.363901226067727	1.8049835836911163	1.2678238460492914
最大值	0.7054136191667166	0.37788992185423737	0.5568070659684774	3.639878377470987	2.9601137046452486	3.0925747234538474
最小值	0.0	0.0	0.0	0.0	0.0	0.0
方差	0.03689458115499113	0.00867425355323771	0.03503005084307183	1.2125640182614736	0.8451696236421472	1.1834527696237325

PSO(没有做限制条件)

平均值	0.1524538925449293	0.4063532168076496	0.16917279341200148	2.3185239531361472	0.5421052098700129	2.013797713974426
最大值	0.34453027375894574	0.7174233696696973	0.37564161234871046	5.4177773708070305	1.1890653113168108	3.5766820951654426
最小值	0.0	0.0	0.0	0.0	0.0	0.0
方差	0.013853155631964514	0.03205463329625125	0.008587606871970228	3.2386447177913764	0.09804638346870778	1.2324683512468377

粗限制

平均值	0.10757137275581333	0.3735228256653741	0.23198212032804535	1.570703075613257	0.8509599926324036	2.6231775355330447
最大值	0.289567581492877	0.7887668797357679	0.42664220584626045	3.9699626984243848	1.9019318021402878	5.027206508904418
最小值	0.0	0.0	0.0	0.0	0.0	0.0
方差	0.005656757390906082	0.040755660590186676	0.020160932330427282	1.9184493038557175	0.2879328538007766	2.0108557282147244

胶道点云中心线提取

根据每条线激光对应点云中点的 y 轴坐标相同，求解 x 轴中值并提取对应点即可。

FRP 实现内网穿透

5 分钟，使用内网穿透快速实现远程桌面 - 知乎 (zhihu.com)

Frp 内网穿透-CSDN 博客

使用 TSAI 和 PSO 分别求解手眼矩阵

近期工作主要是对网上找到的一些关于手眼标定的数据进行求解。使用经典的 TSAI 方法以及粒子群最优化算法。以下是针对用一组数据求解手眼矩阵得出的误差分析，就目前的实验来看，TSAI 和 PSO 算法的最终误差相差不大，后续如果针对具体细节对 PSO 算法进行更进一步的改进，也许会较之现在有明显的提升，此外因为 PSO 算法是求解最优化问题的算法，对于最后的手眼标定矩阵的解并无唯一，相反，TSAI 是专门求解手眼矩阵的算法，最后的求解总是相同的，后续可能需要考虑这方面的因素。

TSAI

平均值	0.28199778570207157	0.20096888177634056	0.2664016288584333	1.363901226067727	1.8049835836911163	1.267823846049
最大值	0.7054136191667166	0.37788992185423737	0.5568070659684774	3.639878377470987	2.9601137046452486	3.092574723453
最小值	0.0	0.0	0.0	0.0	0.0	0.0
方差	0.03689458115499113	0.00867425355323771	0.03503005084307183	1.2125640182614736	0.8451696236421472	1.183452769623

PSO(没有做限制条件)

平均值	0.1524538925449293	0.4063532168076496	0.16917279341200148	2.3185239531361472	0.5421052098700129	2.0137977139744
最大值	0.34453027375894574	0.7174233696696973	0.37564161234871046	5.4177773708070305	1.1890653113168108	3.5766820951654
最小值	0.0	0.0	0.0	0.0	0.0	0.0
方差	0.013853155631964514	0.03205463329625125	0.008587606871970228	3.2386447177913764	0.09804638346870778	1.2324683512468

粗限制

平均值	0.10757137275581333	0.3735228256653741	0.23198212032804535	1.570703075613257	0.8509599926324036	2.6231775355330
最大值	0.289567581492877	0.7887668797357679	0.42664220584626045	3.9699626984243848	1.9019318021402878	5.0272065089044
最小值	0.0	0.0	0.0	0.0	0.0	0.0
方差	0.005656757390906082	0.040755660590186676	0.020160932330427282	1.9184493038557175	0.2879328538007766	2.0108557282147

10.27

Wsl 配置深度学习

2023 最新 WSL 搭建深度学习平台教程（适用于 Docker-gpu、tensorflow-gpu、pytorch-gpu) - 知乎 (zhihu.com)

10.31

1. vscode 配置多个分支(实现多存储库)

使用 VSCode 便捷实现 Git 进阶功能!(包含同一项目配置多个远程 Git 仓库解决方案) ╰(´・ω・)つ——☆✿✿✿ 项目协同_vscode 管理多个推送地址-CSDN 博客

2. 前端预览 DOCX 文件

mammoth.js 预览 word docx 文档 使用示例 demo example