**Day1 --7.22**

明确数据融合主要任务及一些技术路线。

主要任务：

将针对与同一物体的不同的obj格式的文件中存储的坐标数据提取出来，这些数据是不同坐标系下的数据想办法转化到同一个坐标系下，然后拼凑还原出原来的物体并优化。

技术路线：

1. 提取obj文件中的坐标数据
2. 进行ICP配准
3. 对结果进行抽稀优化

**Day2 --7.23**

进一步了解任务细要，并确定要使用PCL库。

在原始的电脑上安装visual studio并删除掉ubuntu系统。

**Day3 --7.24**

尝试在VS2022上安装PCL库：PCL1.14.0安装教程见：

<https://blog.csdn.net/2301_76657565/article/details/137012393>

**Day4 --7.25**

学习PCL教程，见：

<https://github.com/HuangCongQing/pcl-learning/blob/master/PCL%E5%AD%A6%E4%B9%A0%E6%8C%87%E5%8D%97%26%E8%B5%84%E6%96%99%E6%8E%A8%E8%8D%90.md>

**Day5 --7.26**

1、成功使用PCL完成两个obj文件的配准，并将配准后的两个文件合并。

遇到的问题1：调用pcl::io::saveOBJFile函数将点云数据存储为OBJ文件时报错：

解决方法：在合并过程中查阅PCL库官方文档得知PCL1.14.0版本不支持将pcl::cloud保存为OBJ文件，因此需要另写函数进行保存。

在完成配准后，将剩余文件按照设为标准的obj文件（如25.obj）进行坐标系转换，通过Egien库实现，将参与实验的obj文件全部转换到标准obj文件（25.obj）的坐标系下。

遇到的问题2：发现25.obj文件与26.obj文件能很好地进行配准，而25.obj和27.obj、28.obj等扫描得到的点云数据差异过大，不能进行配准。尝试采取相邻两个文件相互配准以此类推的方式，发现27.obj文件与28.obj文件也不能很好地进行配准，代码运行时间过长且未得到结果。

可以尝试的方法：优先选择能进行配准的文件优先进行配准，并且将配准结果与设为标准的obj文件进行合并得到新的obj文件，并以此为新的标准继续进行实验。

**Day6 --7.27**

配准包括粗配准和精配准。粗配准为获得R旋转矩阵（3\*3）和T平移向量（3\*1）的初值，精配准为不断迭代，计算最终的R和T。其中R与α、β、γ三个参数有关，T与tx、ty、tz三个参数有关。配准过程中的坐标变换为

Pt=R\*Ps+T

1. 了解ICP算法的原理和步骤：

ICP算法的目的：将处于不同坐标系下的点，通过最小化配准误差，变换到同一个共同的坐标系中。

ICP算法有四种：

1. ICP：最原始的版本，使用点与点进行匹配
2. NDT：解决ICP直接用点进行计算的速度问题
3. GICP：通用化的ICP，均衡使用点线面的匹配
4. 基于点到面匹配的ICP。

ICP配准的步骤：

1. 对于两个点云，初始化一个变换矩阵，将源点云变换到目标点云的坐标系中去。其中：初始化变化矩阵的方法有：手动估计、利用先验信息（如传感器的初始位姿或者两个点云在某一时间点的相对位置）、粗略配准。 本次实验是采用粗略配准的方法。

其中，粗略配准有：

①基于特征的配准：提取点云中的特征点（如角点，边缘点等），然后通过匹配这些特征点来初步对齐。

②全局配准算法：使用一些全局配准算法（如RANSAC、Super4PCS等）来找到一个初始变换。

1. 最近邻搜索：对于源点云中的每个点，找到目标点云中距离最近的点作为对应点。使用高效的数据结构和算法来加速搜索过程，常用的方法包括KD树（k-Dimensional Tree）和FLANN（Fast Library for Approximate Nearest Neighbors）等。
2. 计算变换：使用对应点对计算出一个最优的刚体变换（包括旋转和平移），使得源点云与目标点云的对应点之间的距离最小。

具体步骤：

①计算对应点的质心： 计算源点云和目标点云对应点的质心（中心点）。

②去质心化： 将对应点对的坐标减去各自的质心，使其中心化。

③计算协方差矩阵： 计算去质心化后的源点和目标点之间的协方差矩阵。

④使用SVD分解协方差矩阵： 对协方差矩阵进行SVD分解，得到旋转矩阵。

⑤计算平移向量： 通过质心和旋转矩阵计算出平移向量。

1. 应用变换：将计算出的变换应用到源点云上，使其更接近目标点云。

5）迭代：重复步骤2到4，直到收敛（误差函数的值达标，或者迭代次数达到预设值）。

**Day7 --7.28**

安装虚拟机和Ubuntu20.04.2。

要在Ubuntu系统上编译C++需要安装编译器，可通过命令行安装g++，gdb并在vscode中安装C/C++扩展。

要在vscode上进行编译需要在项目文件夹下创建一个./vscode文件夹。然后在./vscode文件夹中创建一个task.json文件和一个launch.json文件,然后写一个cpp文件，如main.cpp。

按 Ctrl+Shift+B 编译该项目。VSCode 将使用 tasks.json 中定义的任务来编译 main.cpp 文件，并生成一个名为 main 的可执行文件。两个json文件内部代码见附件tasks.txt和launch.txt。

若设置的cpp文件为hello.cpp，则需要将tasks.json文件中的args参数中的cpp文件的参数改成hello.cpp，同时需要launch.json文件中的program参数中的指向路径必须与tasks.json文件args参数”-o”后的参数（将生成的可执行文件）保持一致

若有多个cpp文件需要配置，则在tasks.json文件中的args参数中的cpp参数后面再添加一项，需要注意的是两个cpp文件必须不能出现同一个函数，类似于同一工程项目中的不同部分。

**Day8 --7.29**

在ubuntu系统上安装PCL库，cmake也需要手动安装，并用vscode编译代码：

1）创建一个项目文件夹，在该文件夹下创建一个src文件夹，并在其中创建一个cpp文件，如main.cpp。

2）需要新增一个CMakeList.txt文件用来配置CMake构建系统，与src文件夹处于同一目录下：

cmake\_minimum\_required(VERSION 3.10)

project(my\_pcl\_project)

# 设置C++标准

set(CMAKE\_CXX\_STANDARD 14)

# 查找PCL包

find\_package(PCL 1.10 REQUIRED)

# 添加源文件

add\_executable(${PROJECT\_NAME} src/main.cpp)

# 链接PCL库

target\_link\_libraries(${PROJECT\_NAME} ${PCL\_LIBRARIES})

# 包含PCL头文件

include\_directories(${PCL\_INCLUDE\_DIRS})

add\_definitions(${PCL\_DEFINITIONS})

1. 在该项目文件夹下创建一个build文件夹
2. 在该项目文件夹下创建.vscode文件夹，并在其中创建tasks.json和launch.json文件，两个json文件的代码见附件tasks-PCL.txt和launch-PCL.txt。（launch.json里的program那一栏的末尾名称必须与CMakeLists.txt文件生成的可执行文件的名称一致）
3. 在.vscode文件夹下创建settings.json文件夹，里面注明一些头文件的路径，可以去除掉main.cpp文件的找不到头文件的报错，也可以创建c\_cpp\_properties.json来去除。(记得核对自己所安装的库的版本)
4. 通过终端导航到该项目文件夹下的build文件夹下，运行cmake .. 指令来生成CMake构建文件。
5. 回到vscode 按住Ctrl+Shift+B在vscode中编译项目。

**Day9 --7.30**

1、学会使用共享文件夹

2、在Ubuntu系统上使用vscode运行之前的代码发现，这样配置好的环境下，运行程序时当前工作目录默认是项目的根目录，而不是源代码文件（cpp文件）

所在的目录，与以前使用visual studio的情况不太一样。例如：此时项目文件夹为code，cpp文件位于code/src/main.cpp，然而工作目录是code/

注：具体的工作目录是在tasks.json和launch.json文件的cwd参数中设置的。

当前工作目录可以通过引入头文件#include<cstdlib> ，并在函数中调用system(“pwd”);查看。

3、重新编写读取和保存OBJ文件的函数：

读取OBJ文件时若要读取面数据，也需要自己定义读取函数，PCL库自带的读取OBJ文件的函数只能读取点数据，不能读取面数据，同时自带的保存为OBJ文件的函数不能直接将PointCloud形式的数据保存为OBJ文件，只能适用于TextureMesh和PolygonMesh形式的数据。

**Day10 --7.31**

1. 在进行ICP配准时发现将两个OBJ文件在设置为源文件和目标文件的时候调换一下，得到的结果就出现了一些差异。目测进行粗略配准设置初始变换矩阵的过程不够严谨，或许需要自行编写点云特征提取的算法来进行粗略配准，得到初始变换矩阵，然后再进行ICP配准。
2. 安装了开源软件cloudcompare，发现其能够自动将放入进去的OBJ文件进行点云拼接，并且处理速度极快，后续打算研究其在点云配准方面的算法。

**Day11~Day13 --8.1~8.3**

1. 学习使用Git

Git是目前世界上最先进的分布式版本控制系统。

Git操作总结：

先git config --global user.name “your name”

再git config --global user.email “your email”

（上面的操作在同一台机器上只需要执行一次）

建一个项目文件夹，然后cd到这个文件夹，再git init 把这个目录变成git可以管理的仓库。

在文件夹中新建文件后可以git add +文件名来上传，git add . 是添加所有文件，之后要用命令

git commit -m + （空格）”本次提交文件的说明”来告诉Git把文件提交到仓库。

这些都是本地仓库，后续需要手动在github上新建一个远程仓库，然后找到链接复制过来，如：git remote add origin <https://github.com/liekaide/test.git>

除此之外可以输入git branch -M main 来修改当前分支的名称，在-M后面输入想要的名称。

查看本地分支：git branch 查看远程分支：git branch -r 两个都查看：git branch -a

创建分支：git branch + 新分支的名称。

切换到新分支：git checkout +新分支名称

创建并切换到新分支：git checkout -b +新分支名称

将新分支推送到远程仓库：git push -u origin +新分支名称

合并分支：先切换到主分支,再执行git merge +要合并的分支名称

删除本地分支：git branch -d +分支名称

删除远程分支：git push origin --delete +分支名称

克隆别人远程仓库上的项目到本地：

git clone +地址

进入新出现的项目文件夹后，删除掉原本的.git文件夹，然后cd到这个文件夹执行：

git init

然后连接到自己的远程仓库：

git remote add origin <https://github.com/liekaide/test.git>

（修改该克隆仓库所连接的远程仓库：git remote set-url origin +地址）

然后可以用：git checkout -b ubuntu origin/ubuntu创建并连接到远程仓库的分支。然后再正常git add .

查看当前本地仓库所连接的远程仓库地址：

git remote -v

删除远程仓库的别名：

git remote remove +仓库别名

产生过时分支时

清理过时的分支引用：

git remote prune origin

再获取远程分支：

git fetch origin

git add 之后，如果觉得上传错了不想继续了，可以用git status查看上传的状态，并用：

git restore --staged +文件名来取消上传。

**Day14 --8.4**

在ubuntu系统上使用git 上传项目到github，同时发现ubuntu系统上上传文件到github仓库，最好使用SSH，不用HTTPS。

换行符有CRLF和LF两种不同的表示方式，在Windows系统上使用CRLF在Linux系统上用的是LF。在进行跨系统文件传输的过程中需要注意换行符的处理，例如在Windows上创建的文本文件在Linux系统上打开时，可能会出现多余的回车符。

常见的解决办法包括使用文本编辑器或IDE的设置来自动处理换行符，或者在脚本和代码中进行统一的转换。

获取到cloudcompare的源码并大致找到配准工作相关的文件。

**Day15 --8.5**

确定不需要再进行配准，现有的OBJ文件可直接用于点云的降噪

记录永久挂载共享文件夹的步骤：

1. 在VMware虚拟机上设置共享文件夹。
2. 进入虚拟机使用sudo mkdir -p /mnt/hgfs创建挂载点
3. 运行sudo vmhgfs-fuse .host:/ /mnt/hgfs -o allow\_other挂载共享文件夹，完成后可在hgfs文件夹中看到我们的共享文件夹。
4. cd ..到初始文件夹，然后cd进入etc文件夹，执行geidt fstab打开一个文件，在末尾添加.host:/ /mnt/hgfs fuse.vmhgfs-fuse allow\_other,uid=0,gid=0,umask=022 0 0（记得检查）
5. 没有权限保存的话可以用sudo nano /etc/fstab 打开再进行修改。
6. 执行ln -s /mnt/hgfs/winshare /home/hj/桌面 将共享文件夹添加到桌面

**Day16 --8.6**

1. 阅读文献：

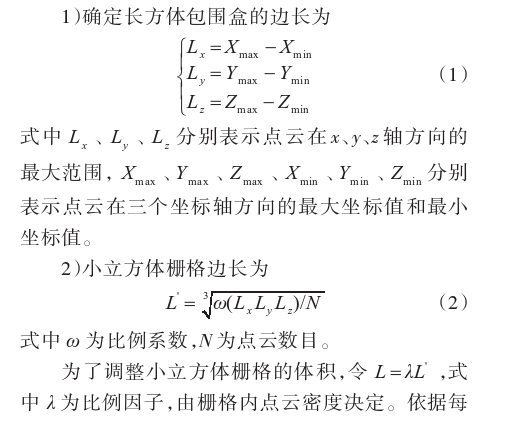
《基于高斯曲率的航空模锻件点云降噪方法》

点云处理一般包括降噪、光顺以及曲面重建等步骤。点云噪声有两大类：1)偏离主体点云且悬浮在周围的稀疏点以及远离主体点云且小而密集的点云；2)与主体点云混

杂在一起的噪声点。

操作步骤：

1. 点云栅格化：根据点云坐标计算出总点云的X、Y、Z坐标的最大和最小值。然后确定长方体的边长，并计算小立方体栅格的边长。



1. 去除第一类噪声点，对各个栅格内的点云数做统计，其数值小于预设阈值的栅格，将其删除。对于点云数大于等于预设阈值的栅格，通过k邻域搜索，遍历所有的点云栅格，确定属于同一片点云的所有栅格，当点云片所包含的点云栅格是最多时，将其保留；剩下的即为偏离主体密集小片点云，将其删除。
2. 去除第二类噪声点：首先将求得k邻域每个点的高斯曲率κi以及所有点的平均高斯曲率κˉ[16],通过比较可知,k邻域内任意点高斯曲率小于平均高斯曲率的点集,属于特征变化小的低频平滑区域,采用中值滤波方法,对其点云进行下采样。公式如下：

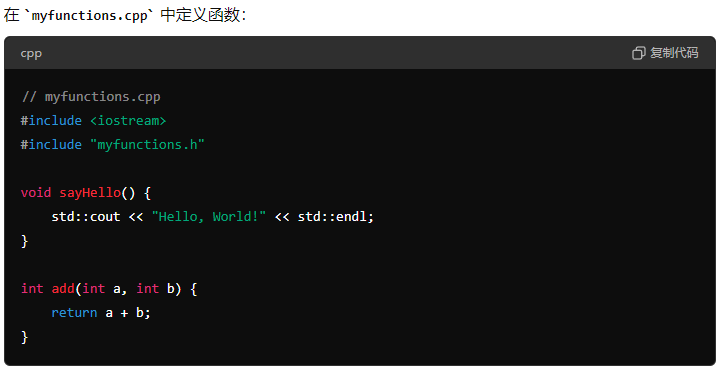
**Day17~18 --8.7~8.8**

1、C++创建函数文件，要先写一个头文件来声明函数，如下：



其中，// myfunctions.h #ifndef MYFUNCTIONS\_H #define MYFUNCTIONS\_H 和#endif // MYFUNCTIONS\_H 属于头文件保护，目的是防止同一个头文件被多次包含，造成重复定义的错误。它的实现方式是在头文件的开头和结尾使用预处理指令 #ifndef、#define 和 #endif。

然后再创建相应的cpp文件来定义函数，如下：



最后，在main.cpp文件中调用这个函数即可，如下：



设置完之后，需要在CMakeList.txt中添加：

# 包含项目的头文件目录

include\_directories(include)

# 添加可执行文件

add\_executable(myprogram src/main.cpp

src/moduleA.cpp

src/moduleB.cpp )

1. 哈希表是一种数据结构，它使用一种称为哈希函数（Hash Function）的映射函数，将键（Key）映射到与之关联的值（Value），可以理解为存储键值对的数组。而哈希表中的数据并不是按照顺序存储的。

**Day19~20 --8.9~8.10**

无

**Day21 --8.11**

计划以单个OBJ文件来测试算法的效果。目前总体的大致思路为：一个一个先平滑一遍，然后想办法合并平滑后的单个OBJ文件，考虑合并后再度进行平滑。同时最需要注意的是合并的过程将原有的特征模糊掉。

补充git：新建一个文件夹存储文件准备上传到远程仓库指定分支时：

1. 先建空文件夹，cd进去，然后git init初始化仓库
2. git remote add origin +地址
3. 随便在文件夹里放一个文件，然后git add .和git commit -m “ ‘
4. git fetch origin 拉取分支
5. git checkout -b ubuntu origin/ubuntu 创建本地分支并连接到远程分支，这一步会将本地仓库与远程仓库同化。
6. git push -u origin 一次
7. 在文件夹里添加好要上传的文件
8. git add .加git commit -m” “
9. Git push -f origin

**Day22 --8.12**

MLS 平滑的核心思想是在每个点的邻域内，通过加权最小二乘法拟合一个曲面（通常是多项式曲面）。然后，将点云中的点投影到拟合的曲面上，以得到平滑后的点云。

上采样是指在点云处理中增加点的数量，使点云更加稠密的过程。简单来说，上采样是在原有点云数据的基础上，生成额外的点，以更好地表示物体的表面或提高点云的分辨率。

体素网格滤波是一种常用的点云数据降采样技术，它通过将点云空间划分为固定大小的立方体（称为体素或Voxel），然后将每个体素中的点简化为一个代表点，从而减少点的数量。

**Day23 --8.13**

可以考虑将OBJ文件存储为mesh而不是pointcloud进行处理。

**Day24 --8.14**

考虑研究泊松重构来完成补洞工作。

**Day25~Day27 --8.15-8.17**

Have a rest.

**Day28 --8.18**

**1、C++学习 --结构体和类**

结构体的成员函数可以直接访问结构内部的成员变量，直接在结构体内部定义时不需要设置相应的形参，只需要设置需要输入的其他形参。

类的成员变量也叫属性，类的成员函数也叫方法/行为，类的成员函数可以定义在类的外面。

enum枚举，定义一个枚举类可以设置一个东西的可选择项来提高代码的可读性和维护性。例如：

// 定义一个枚举类型，表示游戏状态

enum class GameState {

START, // 游戏开始

IN\_PROGRESS,// 游戏进行中

GAME\_OVER // 游戏结束 };

赋值和初始化的时候可以使用GameState state = GameState::START;

非静态成员：每个对象有自己的副本，通过对象实例访问，生命周期与对象一致。（一般的成员变量都是非静态）

静态成员：所有对象共享一份数据，可以通过类名访问，生命周期与程序一致。（static开头）

访问静态成员、枚举类、命名空间成员时用：：

访问非静态成员用.

在类的声明中定义的函数都将自动成为内联函数，在类的声明之外定义的函数不是。

**Day29 --8.19**

1. **学习C++**

**1.1 --向量**

确定向量长度用size\_t=vec.size()，确定字符串长度用.length()，确定数组长度用sizeof(arr)/sizeof(arr[0])。

**1.2 --二分查找**

设定左边和右边的索引，左边为0，右边为数组长度减一，两者相加除以二算出中间值的索引。然后将中间值与要查找的值进行比较，看看在哪个半区，左边的话就将右边的索引设置为中间索引mid-1，然后重新计算mid，结束的条件为left>right。算法复杂度log2n。

**Day30 --8.20**

1. 学习C++

使用if语句遇到数组越界情况，可以使用if(i+1<len && s[i+1]==’L’)

1. 曲面重建

**Day31 --8.21**

1. 学习C++

**Day32 --8.22**

**1、学习C++**

n>>j是二进制右移：

int n = 8; // 二进制: 1000

int j = 2;

int result = n >> j; // 结果: 2 (二进制: 0010)

**Day --8.26**

**1、学习C++**

vector<Employee\*> employees 中间的Emplyoee是一个类时，还是可以用.size()来获取这个向量的长度。

访问类的成员变量或成员函数时，使用 . 或 -> 取决于你是通过对象还是通过指针来访问这些成员。当你有一个类的对象时，使用 . 来访问成员变量或成员函数。当你有一个指向类对象的指针时，使用 -> 来访问成员变量或成员函数。

1. **项目**

确定重建的过程需要先计算法向量，之前的代码可以修改参数继续用。

**Day --8.27**

1. **C++算法题**

Vector可以用find(vec.begin(),vec.end(),value)!=vec.end()；来查找元素是否存在于向量中。也可以用count(vec.begin(),vec.end(),value)来数元素在向量中出现的次数。