



北京航空航天大學  
BEIHANG UNIVERSITY

# 3D点云数据处理和可视化实践(入门)

## Lecture2-三维点云配准机制

国新院实验实践课  
工程师通用技术科教平台



教师：欧阳真超



邮箱：ouyangkid@buaa.edu.cn



学期：2024年秋季

# 目录

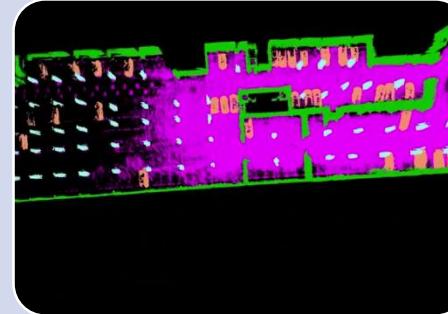
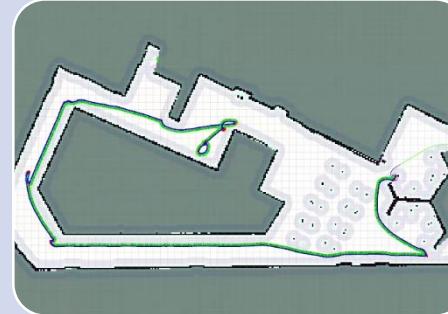
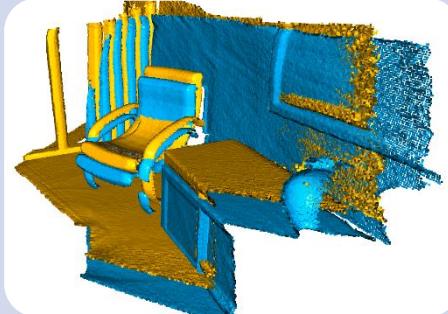
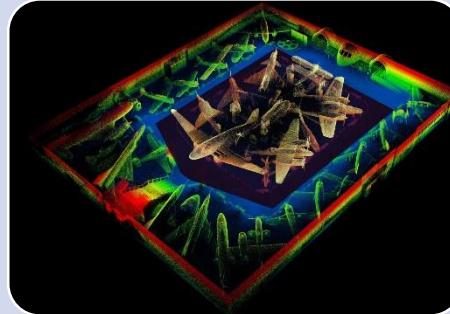
## Contents

- 01 点云配准
- 02 课程目标
- 03 学习目标
- 04 实验设备及数据
- 05 实验

# Part 1 | 课程内容安排

课程内容

- 课程围绕**三维点云数据**的处理和可视化实践的**4个核心内容**开展，具体包括



三维点云计  
算基础

(1 + 3学时)

三维点云配  
准机制

(1 + 3学时)

同步定位与  
建图

(1 + 3学时)

点云语义分  
割模型

(1 + 3学时)

# Part 1 | 课程内容安排

课程安排

## » 三维点云计算基础

理论教学-1课时

- 传感器原理
- 数据获取
- 数据结构
- 可视化
- Open3D API
- Realsense传感器

实践教学-3课时

- 开发环境安装
- 传感器数据获取
- 数据存储与播放
- 基础运算
- 可视化

## » 同步定位与建图

理论教学-1课时

- 机械激光雷达
- 经典SLAM流程
- ROS
- SLAM评估机制

实践教学-3课时

- ROS基本命令操作
- Bag回放和数据读取
- 典型算法运行
- 可视化和测试

## » 三维点云配准机制

理论教学-1课时

- 配准机制
- 李群代数
- ICP配准流程
- CICP配准流程

实践教学-3课时

- 传感器数据采集
- ICP配准实践
- CICP配准实践
- 连续配准和可视化

## » 点云分割

理论教学-1课时

- 语义分割任务定义
- 深度学习典型框架
- 典型数据表征介绍
- 分割量化评估

实践教学-3课时

- 基于SLAM的点云语义标注
- 模型训练和测试
- 量化评估和可视化

# 目录

## Contents

- 01 点云配准
- 02 课程目标
- 03 学习目标
- 04 实验设备及数据
- 05 实验

# Part 2 | 课程目标

## » 3D点云数据处理和可视化实践

### 目标学生

- 电子信息和交通运输



#### 先进传感器

激光雷达作为近年来普及的传感器被广泛应用于三维测量和机器人领域。

!



#### 智能机器人

面向先进机器人移动感知建模，在机载和车载领域的理论与实践技术开展介绍。

!

### 学习内容

- 多学科交叉的实践学习



#### 人工智能

深度学习在三维点云数据的应用技术、从“数据-建模-量化评估”的闭环流程开发。

!



#### 编程实践

课程内容基于近年实验室研究的工程经验积累，前沿工具和算法实用性强。

!

- 掌握传感器前沿技术
- 学习激光雷达特性和选型
- 理论与应用相结合

- 机器人类人空间认知
- 智能感知前沿技术剖析
- 通过先进技术吸引学生科研兴趣

- 平台和目标导向
- 感知系统设计
- 基本覆盖神经网络任务流程开发
- 基础算法和模型框架学习

- 软硬件环境操作学习
- 算法理论+编程实践
- 开发流程：多种工具的串联使用
- 可视化（定性）和量化评估（定量）

# 目录

## Contents

- 01 点云配准
- 02 课程目标
- 03 学习目标
- 04 实验设备及数据
- 05 实验

# Part 3 | 学习目标

## » 3D点云数据处理和可视化实践入门

### 1. 两类传感器（硬件）

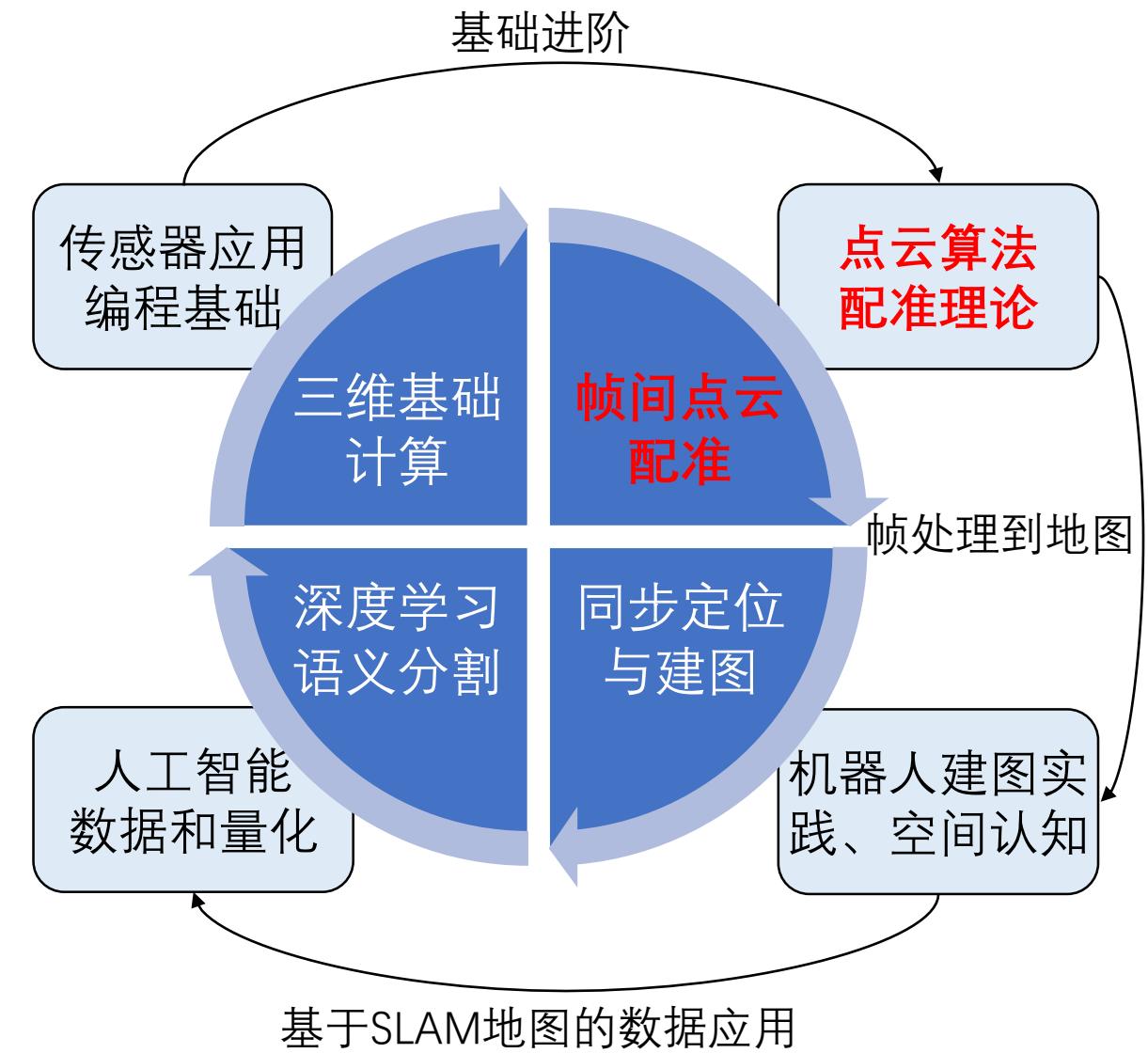
- Realsense: 双目+结构光，稠密视场点云
- LiDAR: 机械ToF测距，稀疏环视点云

### 2. 三维空间表征与计算（理论编程）

- Open3D: 先进算法库使用
- SLAM经典算法流程
- 深度学习: Pytorch+Spcov
- 李群代数, 图优化

### 3. 软件系统（操作实践）

- Ubuntu: shell
- ROS机器人操作系统
- PointLabeler



# Part 3 | 学习目标

## » 3D点云数据处理和可视化实践入门

- 实践课作业以两人为一组，提交可以运行的代码+数据到本人邮箱：  
[ouyangkid@buaa.edu.cn](mailto:ouyangkid@buaa.edu.cn)
- 压缩包注明实验所采用的传感器型号、以及本地执行完成后的屏幕截图。
- 姓名，学号。
- 打包形式: name1\_no.1\_name2\_no.2\_sensortype.zip
- 课程1作业（以子1234文件夹的形式组织）：
  - 基于align-depth2color.py，通过添加可控opencv的bar，调节深度过滤阈值。
  - 基于opencv\_pointcloud\_viewer.py，通过键盘按键存储当前点云帧为.pcd文件，包括xyz rgb。要求采用ascii编码的数据压缩形式。
  - 提交一个>5s的bag，以及对图像和深度信息流同时播放的UI。
  - 对本地采集的pcd进行点云基础操作，提供执行的每个操作函数后的截图。

# 目录

## Contents

- 01 点云配准
- 02 课程目标
- 03 学习目标
- 04 实验设备及数据
- 05 实验

# Part 4 | 硬件介绍

## » 3D点云数据处理和可视化实践



传感器和移动机  
器人平台



内网远程访问的  
服务器和显卡



开源数据、本地  
采集数据和软件

# 目录

## Contents

- 01 点云配准
- 02 课程目标
- 03 学习目标
- 04 实验设备及数据
- 05 实验

# Part 5 | Realsense 点云配准

## » 双目结构光相机

传感器构成:

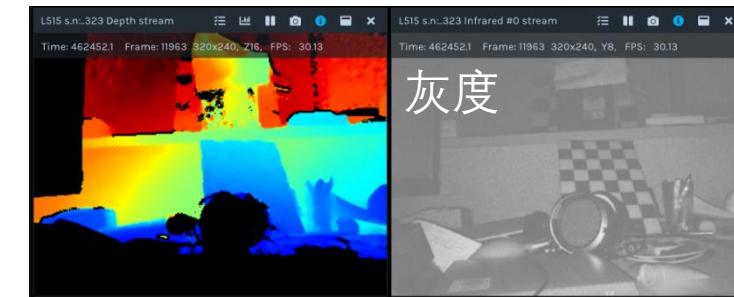
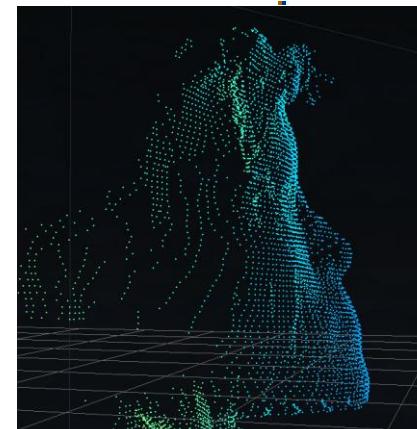
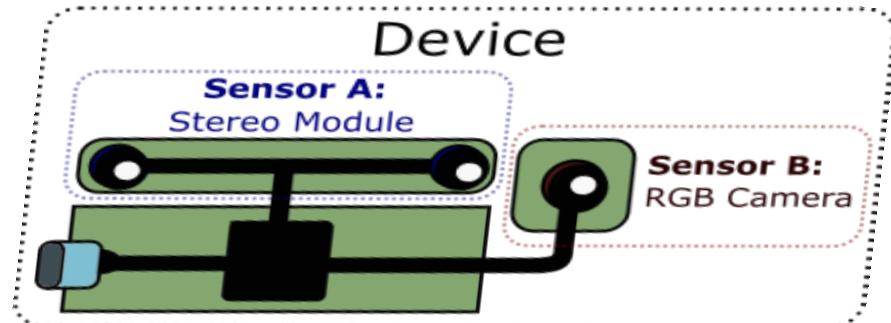
- 数据接口: USB 3.0
- 相机: 单目/双目
- 测距: 结构光, 激光ToF
- 惯性单元: 可选

[传感器选型对比pdf](#)

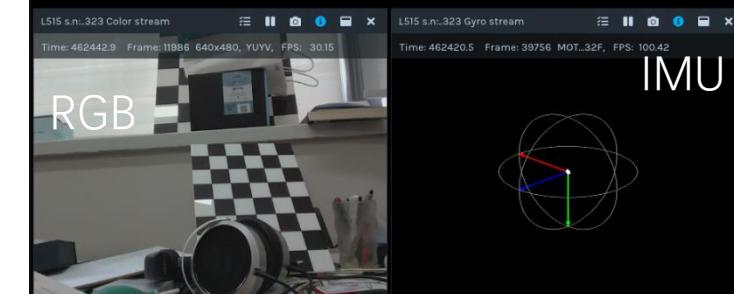
结构光测距相机



RS Intel.RealSense.Viewer.exe



深度



RGB

IMU



3D视角

传感器介绍  
2D视角

# Part 5 | Realsense 点云配准

环境安装

## » 开发环境配置

- IDE: pycharm

<https://www.jetbrains.com/pycharm/download/?section=window>

选择 PyCharm Community Edition

- Python虚拟环境管理: anaconda

<https://www.anaconda.com/download/success>

- 源替换(国内加速):

- <https://mirror.tuna.tsinghua.edu.cn/help/anaconda/> conda清华源
  - conda config --set show\_channel\_urls yes # 生成 .condarc
  - <https://mirror.tuna.tsinghua.edu.cn/help/pypi/> pip源

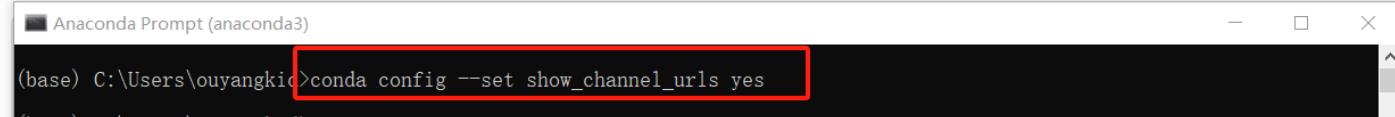
## Anaconda 镜像使用帮助

Anaconda 是一个用于科学计算的 Python 发行版，支持 Linux, Mac, Windows, 包含了众多流行的科学计算、数据分析的 Python 包。

Anaconda 安装包可以到 <https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/> 下载。

TUNA 还提供了 Anaconda 仓库与第三方源 (conda-forge、msys2、pytorch等, [查看完整列表](#), 更多第三方源可以前往[校园网联合镜像站](#)查看) 的镜像, 各系统都可以通过修改用户目录下的 `.condarc` 文件来使用 TUNA 镜像源。Windows 用户无法直接创建名为 `.condarc` 的文件, 可先执行 `conda config --set show_channel_urls yes` 生成该文件之后再修改。

注: 由于更新过快难以同步, 我们不同步 `pytorch-nightly`, `pytorch-nightly-cpu`, `ignite-nightly` 这三个包。



```
Anaconda Prompt (anaconda3)
(base) C:\Users\ouyangkic>conda config --set show_channel_urls yes
```



## Realsense github:

<https://github.com/IntelRealSense/librealsense/tree/master>

# Part 5 | Realsense 点云配准

## 环境安装

### » 开发环境配置

- conda虚拟环境创建 ([conda常用命令](#), 或者 conda -h)

conda create -n open3d python=3.8 # 创建一个名为 open3d的python3.8环境

- 每次通过 activate/deactivate 进入/退出相应的环境

conda activate open3d

pip install **open3d, opencv-python, pyrealsense2** # 当前时间最新版本0.18.0

numpy==1.24.0 # 2.X版本有memory的bug

安装完成后 pip list 可以看到安装完成的相应包，并且可以在程序中进行import操作。

```
nptyping           1.4.4
numpy              1.24.3
open3d             0.18.0
opencv-contrib-python 4.9.0.80
opencv-python       4.9.0.80
```

```
import json
import os
import numpy as np
import cv2
import time
import open3d as o3d
```

绝大多数linux和开发环境自带操作说明: -h/-help查询

Until [2024/8/20](#) tested!

Supported Python versions:

- 3.8
- 3.9
- 3.10
- 3.11

Supported operating systems:

- Ubuntu 18.04+
- macOS 10.15+
- Windows 10+ (64-bit)

验证基础环境配置完成

```
PS H:\中法\2024\研究生课程\三维点云处理和可视化实践\code\lesson1> python -c "import open3d as o3d; print(o3d.__version__)"
0.18.0
```

# Part 5 | Realsense 点云配准

点云数据

## » 传感器数据获取-点云数据结构

.pcd文件

RGB=829K, depth=423K



PCD	write_ascii	compressed	文件大小
1			13.8M
2	√		48.5M
3	√	√	48.5M
4		√	6.09M

Format	Description
xyz	Each line contains [x, y, z], where x, y, z are the 3D coordinates
xyzn	Each line contains [x, y, z, nx, ny, nz], where nx, ny, nz are the normals
xyzrgb	Each line contains [x, y, z, r, g, b], where r, g, b are in floats of range [0, 1]
pts	The first line is an integer representing the number of points. The subsequent lines follow one of these formats: [x, y, z, i, r, g, b], [x, y, z, r, g, b], [x, y, z, i] or [x, y, z], where x, y, z, i are of type double and r, g, b are of type uint8
ply	See <a href="#">Polygon File Format</a> , the ply file can contain both point cloud and mesh data
pcd	See <a href="#">Point Cloud Data</a>

```
o3d.io.write_point_cloud(file_name, pcd, write_ascii=False, compressed=False, print_progress=True)
```

```
o3d.io.read_point_cloud(filename, format='auto', remove_nan_points=False, remove_infinite_points=False, print_progress=False)
```

# Part 5 | Realsense 点云配准

PCD文件

## » 传感器数据获取

PCD文件格式一般包括：

1. Version 版本信息
2. Fields 文件头
3. Size 大小
4. Type 类型
5. Count 元素数量
6. Width 宽度
7. Height 高度
8. Viewpoint 视点信息
9. Points 点规模
10. Data 真数据部分

ASCII编码

```
1 # .PCD v0.7 - Point Cloud Data file format
2 VERSION 0.7
3 FIELDS x y z rgb
4 SIZE 4 4 4 4
5 TYPE F F F F
6 COUNT 1 1 1 1
7 WIDTH 907162
8 HEIGHT 1
9 VIEWPOINT 0 0 0 1 0 0 0
10 POINTS 907162
11 DATA binary
12 荷N窟\xF5>\x93斂廻\x89鯀N窟\xF5>\x93斂廻\x89EON繙s\xF5
Tcm繙s\xF5>Z斂儕\x87 L窟\xF5>\x93斂儕\x87玲L咯恤>削
.x\xF4>欠標涼\x89 J刻黧>紅標鑑\x8A億J刻黧>紅標所\x8
H筐\xF3> 掣噴\x89\xC9H筐\xF3> 掣噏\x8A\x9A-H筐\xF1
H繖鑑>鳶標噏\x8AAR糊繖鑑>鳶標塗\x8CE朏刻黧>紅標墘\x8
讚繅墜F#G纏汎>
```

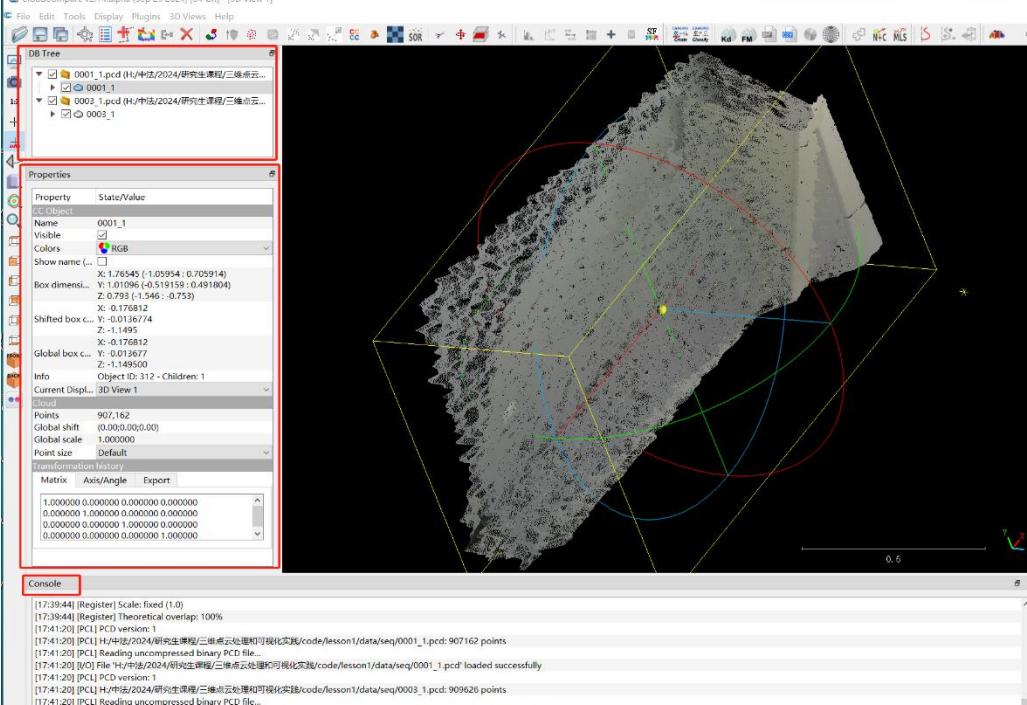
非ASCII编码

```
1 # .PCD v0.7 - Point Cloud Data file format
2 VERSION 0.7
3 FIELDS x y z rgb
4 SIZE 4 4 4 4
5 TYPE F F F F
6 COUNT 1 1 1 1
7 WIDTH 907162
8 HEIGHT 1
9 VIEWPOINT 0 0 0 1 0 0 0
10 POINTS 907162
11 DATA ascii
12 -0.8070340818 0.4785680425 -1.156999946 1.263151315e-38
13 -0.8057702106 0.4785680425 -1.156999946 1.263151315e-38
14 -0.8058970811 0.479395338 -1.159000039 1.253931752e-38
15 -0.8046310251 0.479395338 -1.159000039 1.244712189e-38
16 -0.8033649692 0.479395338 -1.159000039 1.244712189e-38
17 -0.8007117262 0.4785680425 -1.156999946 1.244712189e-38
```

# Part 5 | Realsense 点云配准

## » 传感器数据获取

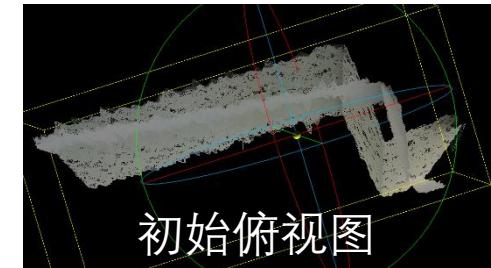
CC配准 (上一次的数据)



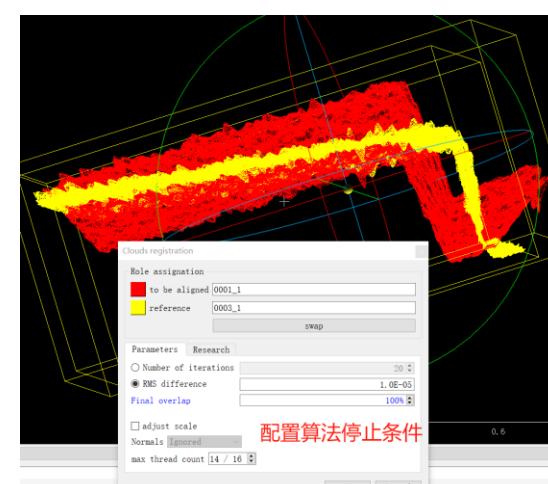
Cloudcompare: <https://www.cloudcompare.org/> 2.14

配准功能:

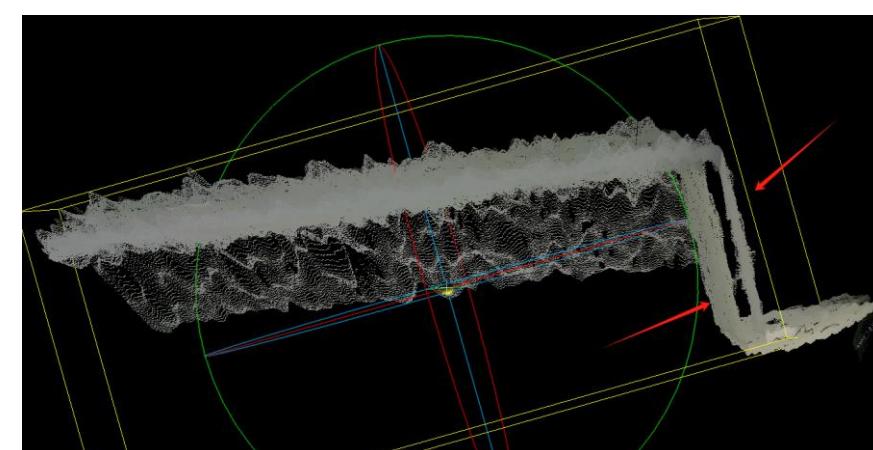
1. 导入两帧点云，需要有一定的共视场 (在图像配准里面为 同名点)
  1. 先测试 frame1 和 frame2: 失败
  2. 再测试frame1 和 frame3: 成功
2. 在DB Tree选择目标点云文件
3. Tools-Registration-ICP
4. 执行并等待



初始俯视图



匹配查看PCD文件信息，并且调整相关属性。



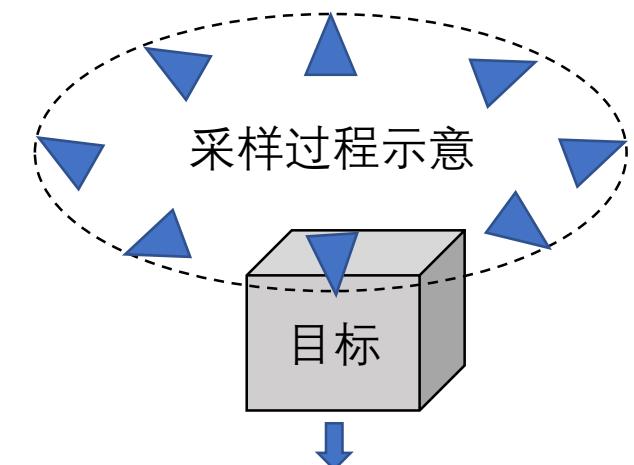
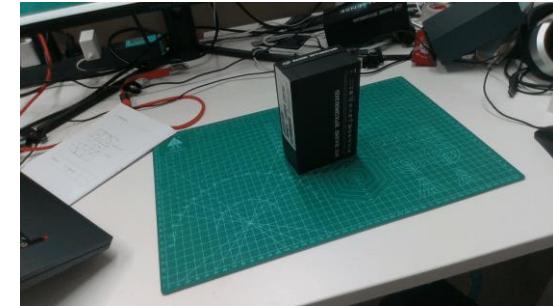
# Part 5 | Realsense 点云配准

数据采集

## » 传感器数据获取

★实验：利用传感器对目标（任意手边的无反光、不透明目标）进行连续环绕拍摄 (>20帧)，并存储 pcd 文件。注意保证帧间视场重叠。

- 使用 0\_prepare\_data.py 进行数据获取，以环绕拍摄的方式(多视角)，围绕目标进行间隔采样。
- 然后将数据存储为XYZRGB的点云形式。
- 本节课将不提供源代码，请基于上一节课的代码完善相关功能。



Final 3D point cloud: GT.pcd



# Part 5 | Realsense 点云配准

CC配准

## » Cloudcompare 处理 (开源软件处理)

- 开启CC
- 导入间隔交大的两帧点云（单帧无全局映射）
- DB Tree选择点云
- Tools
- Registration
- Fine Registration (ICP)
- Ok

Final RMS\*: 0.106106 (computed on 50000 points)  
(\* RMS is potentially weighted, depending on the selected options)

Transformation matrix

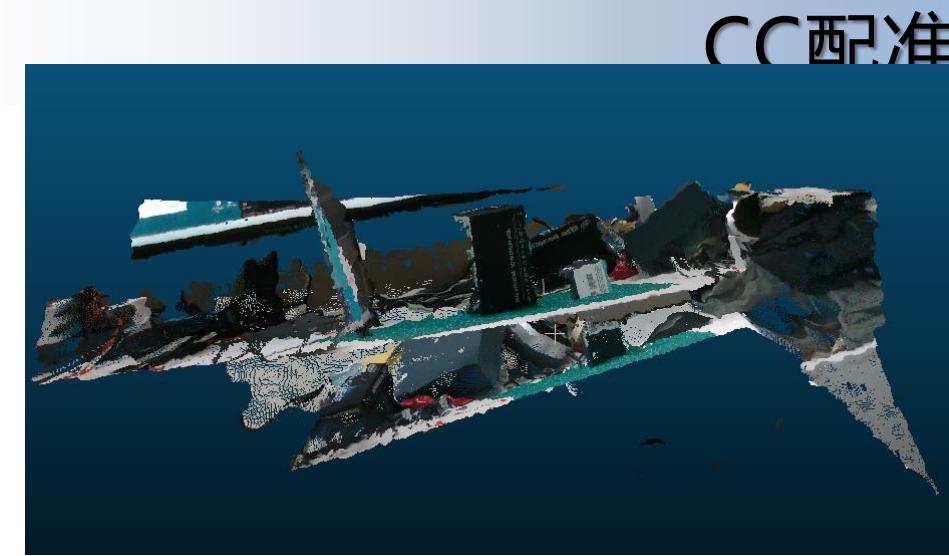
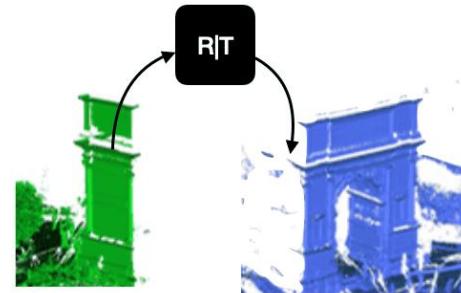
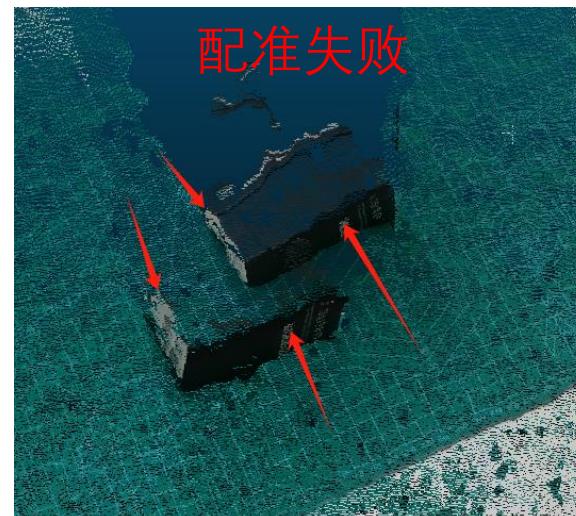
0.965	-0.227	-0.134	-0.124
0.220	0.973	-0.068	-0.075
0.146	0.036	0.989	0.087
0.000	0.000	0.000	1.000

Scale: fixed (1.0)

Theoretical overlap: 100%

This report has been output to Console (F8)

RT矩阵

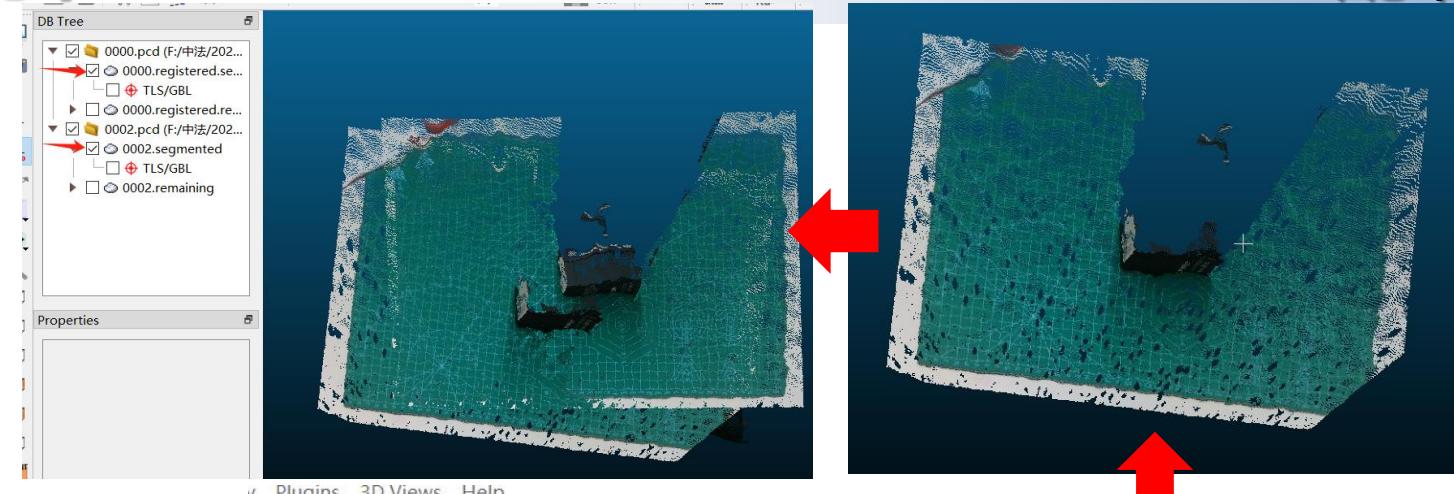


# Part 5 | Realsense 点云配准

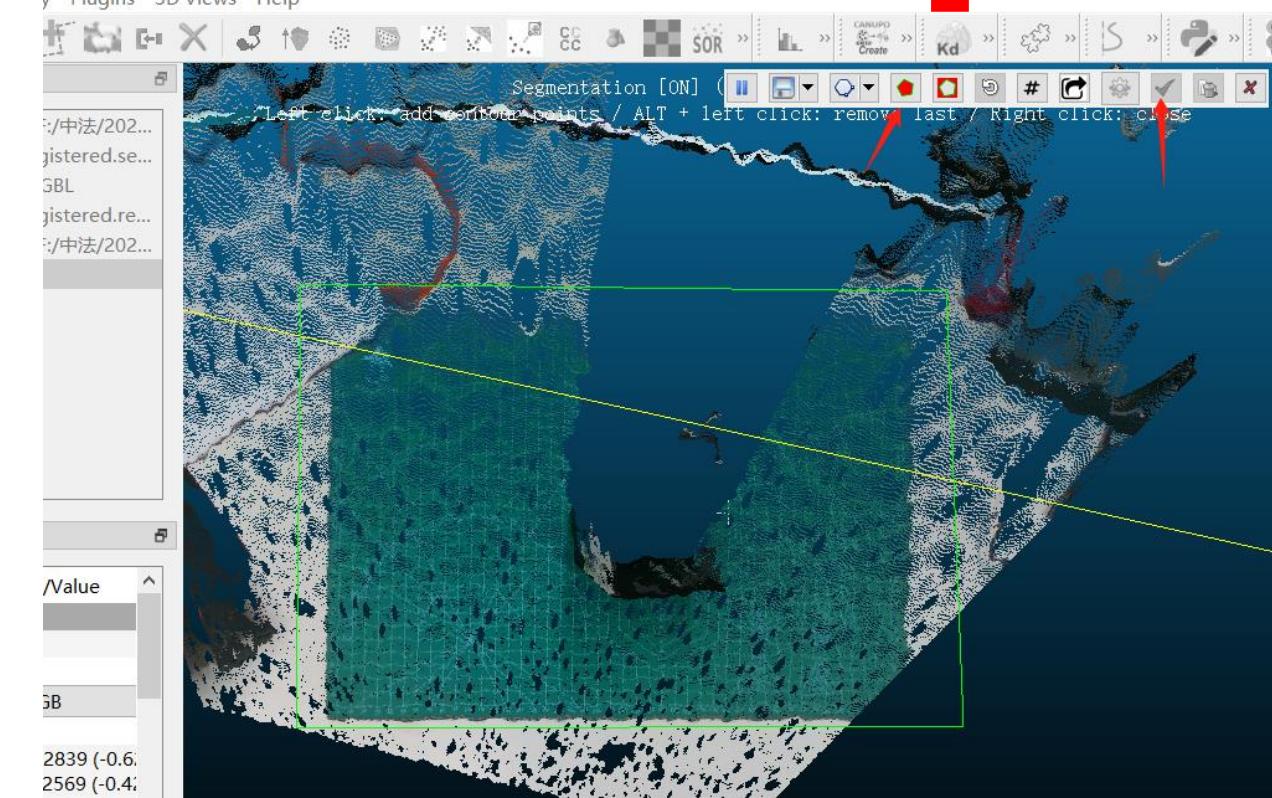
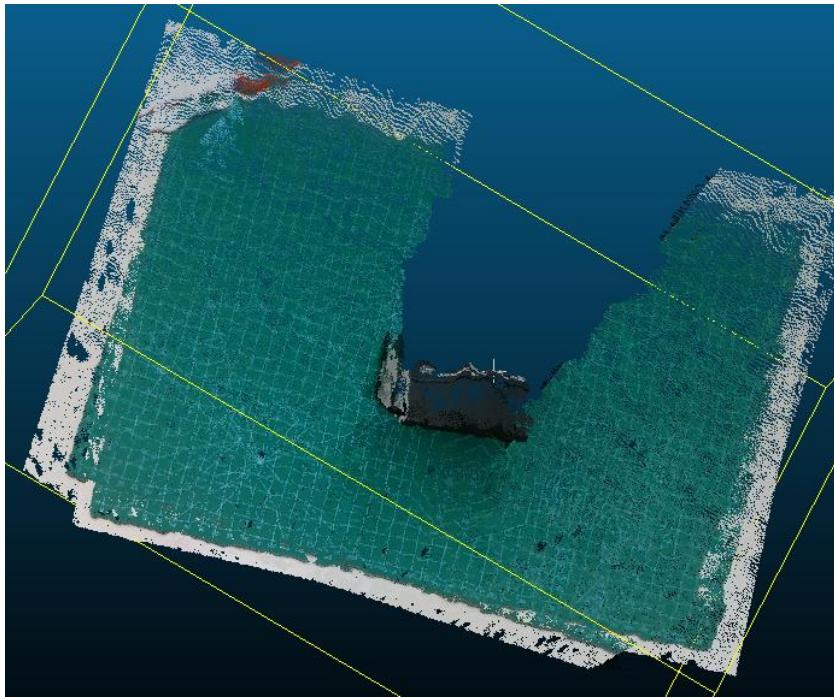
CC配准

## » Cloudcompare 处理

- 选则点云1, 截切工具
- 剔除多余背景
- 重新进行配准



配准效果略微提升



# Part 5 | Realsense 点云配准

刚体运动

## » 点云平移&旋转

理论部分参考李群代数Li Group

```
pcd_tx1 = copy.deepcopy( pcd ).translate( (0.1, 0, 0) ) # meter  
pcd_tx1.paint_uniform_color([0.1, 0, 0]) # as red
```

```
pcd_tx2 = copy.deepcopy( pcd ).translate( (0.2, 0, 0) ) # meter  
pcd_tx2.paint_uniform_color( [0.2, 0, 0] ) # as red
```

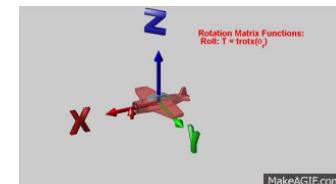
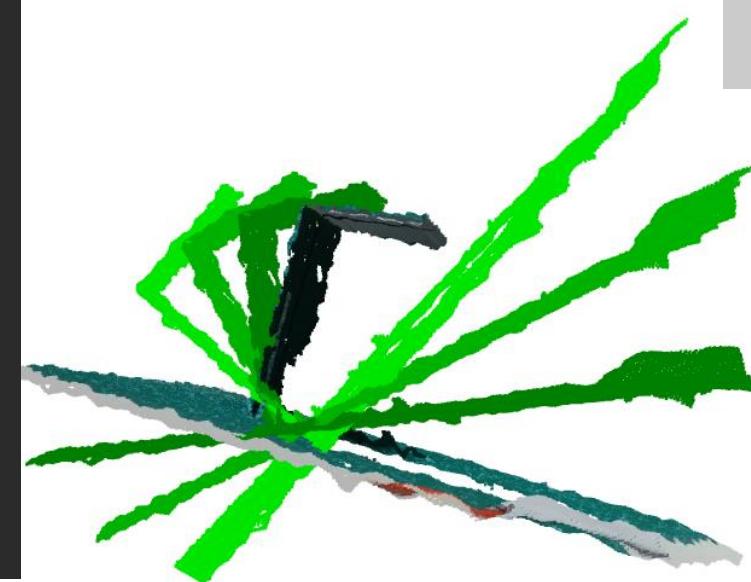
```
pcd_tx3 = copy.deepcopy( pcd ).translate( (0.3, 0, 0) ) # meter  
pcd_tx3.paint_uniform_color( [0.3, 0, 0] ) # as red
```

```
R1 = pcd.get_rotation_matrix_from_xyz((1 * np.pi / 8, 0, 0)) # 45 degree  
pcd_rx1 = copy.deepcopy(pcd).paint_uniform_color([0, 0.5, 0]) # as green  
pcd_rx1.rotate(R1, center=(0, 0, 0))
```

```
R2 = pcd.get_rotation_matrix_from_xyz((2 * np.pi / 8, 0, 0)) # 90 degree  
pcd_rx2 = copy.deepcopy(pcd).paint_uniform_color([0, 0.7, 0]) # as green  
pcd_rx2.rotate(R2, center=(0, 0, 0))
```

```
R3 = pcd.get_rotation_matrix_from_xyz((3 * np.pi / 8, 0, 0)) # 135 degree  
pcd_rx3 = copy.deepcopy(pcd).paint_uniform_color([0, 0.9, 0]) # as green  
pcd_rx3.rotate(R3, center=(0, 0, 0))
```

每次平移0.1m, 颜色变浅  
Rgb数值越大越浅

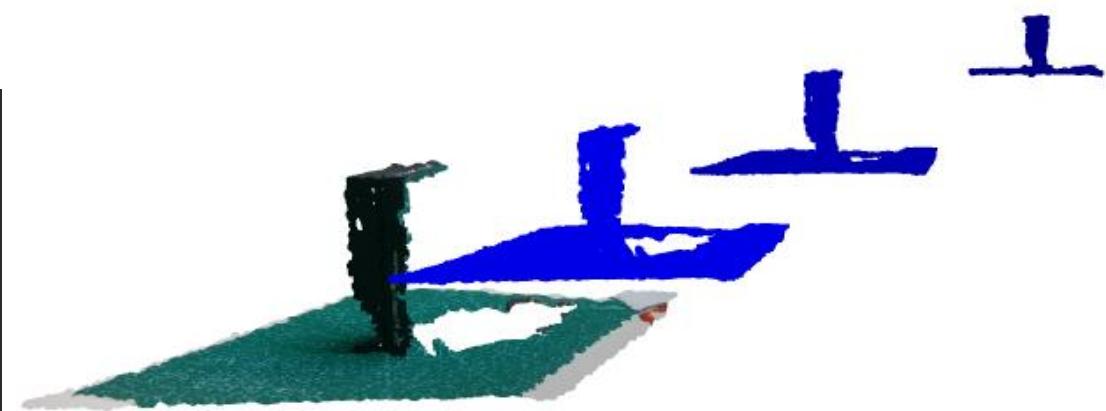


# Part 5 | Realsense 点云配准

刚体运动

## » 点云放缩

```
pcd_s1 = copy.deepcopy(pcd).translate((0.6, 0, 0)) # 0.6  
pcd_s1.scale(0.3, center=pcd_s1.get_center()) # as blue  
pcd_s1.paint_uniform_color([0, 0, 0.5])  
  
pcd_s2 = copy.deepcopy(pcd).translate( (0.4, 0, 0)) #  
pcd_s2.scale( 0.5, center=pcd_s2.get_center() ) # as blue  
pcd_s2.paint_uniform_color([0, 0, 0.7])  
  
pcd_s3 = copy.deepcopy(pcd).translate( (0.2, 0, 0)) # rate=0.2  
pcd_s3.scale( 0.7, center=pcd_s3.get_center() ) # as blue  
pcd_s3.paint_uniform_color([0, 0, 0.9])
```

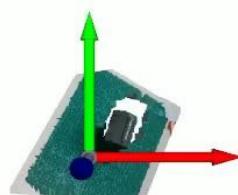


实际上，由于点云主要是现实的精确测量，物理世界不存在仿射变换。真实世界不像图像，存在特征放缩的情形。

## 刚体运动

```
T = np.eye(4)  
T[:3, :3] = pcd.get_rotation_matrix_from_xyz((roll, pitch, yaw)) # R  
T[:, :3] = np.asarray([x, y, z]).T  
PCD.transform(T)
```

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



将旋转和平移组建成RT矩阵，然后进行转换

[R , t  
0 , 1]

### » 旋转不可解耦

★实验：同时绕xyz轴旋转30度（1次旋转），与依次绕x-y-z旋转30度（3次旋转），结果是否一致？



[1 2 0]=y-z-x 蓝色

★实验：同时沿xyz轴平移0.3（1次性），与依次沿x-y-z平移0.3（3次平移），结果是否一致？

#### 1. 旋转顺序：

- 三维空间中的旋转不满足交换律，即绕不同轴的旋转顺序会影响最终的旋转结果。
- 如果一次性绕  $x$ ,  $y$ , 和  $z$  轴旋转 30 度，可以表示为一个复合旋转。这种情况下，最终的旋转状态是同时应用三个轴的旋转。
- 如果依次沿三个轴旋转，每次都绕一个轴旋转 30 度，结果依赖于旋转的顺序。

#### 2. 旋转矩阵：

- 一次性旋转的旋转矩阵可以表示为：

$$R = R_x(30^\circ) \cdot R_y(30^\circ) \cdot R_z(30^\circ)$$

- 而依次旋转的结果则取决于每一步的旋转：

$$R' = R_z(30^\circ) \cdot R_y(30^\circ) \cdot R_x(30^\circ)$$

- 由于矩阵乘法不满足交换律， $R$  和  $R'$  一般不会相等。

# Part 5 | Realsense 点云配准

## KD-Tree

### » KD-Tree

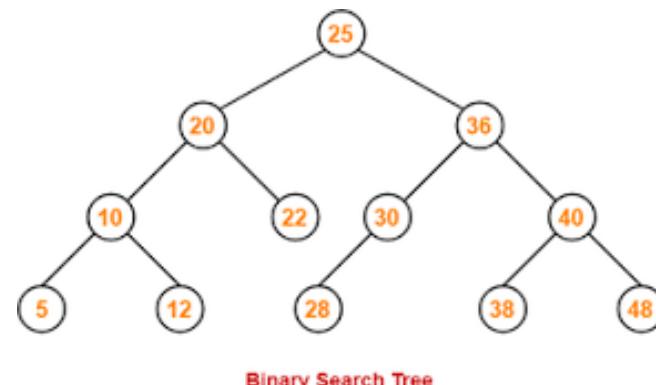
KD-Tree 是一种空间划分数据结构，旨在高效地存储和检索 K 维空间中的数据点。它通过递归地将数据划分为 K 维超矩形区域，形成二叉树的结构，每个节点代表一个数据点并包含以下信息：

- **分割维度**：每个节点通过选择一个维度来分割点集，该维度通常在每次分割时轮换（例如，第一层使用第一个维度，第二层使用第二个维度，依此类推），以确保树的平衡。
- **分割点**：在所选维度上找到中位数值作为分割点，将点集分为左子集和右子集。
- **子节点**：每个节点的左子树包含所有小于分割点的点，右子树包含所有大于或等于分割点的点。

Key	List
3	6 4 1 9 7 3 2 8
3	6 4 1 9 7 3 2 8
3	6 4 1 9 7 3 2 8
3	6 4 1 9 7 3 2 8
3	6 4 1 9 7 3 2 8
3	6 4 1 9 7 3 2 8

$O(n)$

不具备有效的空间结构



$O(\log n)$

• **快速最近邻搜索**：KD-Tree 可以显著加速最近邻搜索。在高维空间中，线性搜索需要检查所有点，时间复杂度为  $O(n)$ ，而 KD-Tree 可以将查询时间缩短至  $O(\log n)$ ，通过有效地利用空间结构进行剪枝。

• **范围查询**：KD-Tree 可以快速找到落在特定超矩形范围内的所有点，而线性方法需要遍历所有点。

空间复杂 换 时间复杂度：构建过程

# Part 5 | Realsense 点云配准

## KD-Tree

### » KD-Tree

- KD-tree的构建

1\_kd-tree.py

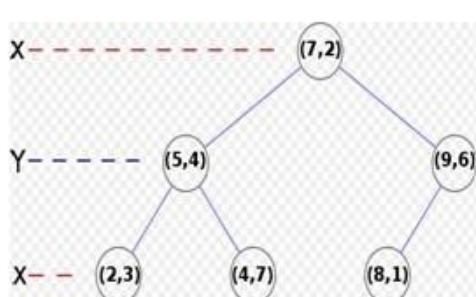


图 1. 2D KD-tree

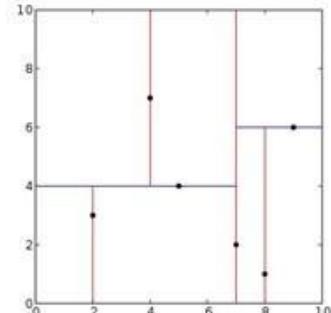


图 2. 2D KD-tree 对应的平面的切割

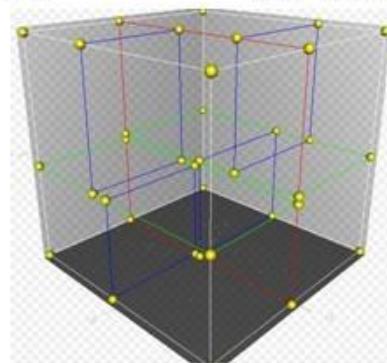


图 3. 3D KD-tree 对应的空间切割 (X-、Y-、Z-aligned)

### 构建步骤:

- 在K维数据集合中选择具有**最大方差**的维度k，然后在该维度上选择**中值**m为切分点对该数据集合进行划分，得到两个子集合；同时创建一个树结点node，用于存储；
- 对两个子集合重复（1）步骤的过程，直至所有子集合都不能再划分为止；如果某个子集合不能再划分时，则将该子集合中的数据保存到叶子结点（leaf node）。

对于拥有n个已知点的KD-Tree，其复杂度如下：

1. 构建:  $O(\log_2 n)$

2. 插入:  $O(\log n)$

3. 删除:  $O(\log n)$

4. 查询:  $O(n^{(1-1/k)} + m)$  # 搜索的m个近邻

希望大家能够在点云处理的基础上，理解三维空间数据的计算效率改进方法，并且能够在将来实践中遇到相应情况，能够想到KD-tree。当然还有octree，由于时间关系，可以自行研究。

### » KD-Tree

- KD-tree其实是K-dimension tree的缩写，是对数据点在k维空间中划分的一种数据结构。Kd-tree是一种平衡二叉树。

```
0: [5 7 6]
1: [5 4 6]
2: [4 8 9]
3: [3 7 5]
4: [5 9 0]
5: [0 0 8]
6: [7 8 9]
7: [7 4 7]
8: [1 6 1]
9: [9 5 4]
Tree structure:
      ____1_____
      /         \
     --4--       3
    /   \     / \
   8     9   0   2
  / \   /
 7   5   6
```

#### 代码流程：

1. 计算方差
2. 基于方差排序
3. 取中位数 → 大约10行代码。
4. 划分子树
5. 添加当前节点到树
6. 遍历左子树
7. 遍历右子树

#### 示例：

假设有六个二维数据点 = {(2,3),(5,4),(9,6),(4,7),(8,1),(7,2)}, 数据点位于二维空间中。为了能有效的找到最近邻，Kd-tree采用分而治之的思想，即将整个空间划分为几个小部分。六个二维数据点生成的Kd-树如图1所示。推广到三维空间的结果如图3所示。

★实验：编程实现kd-tree构建，并可打印树结构。

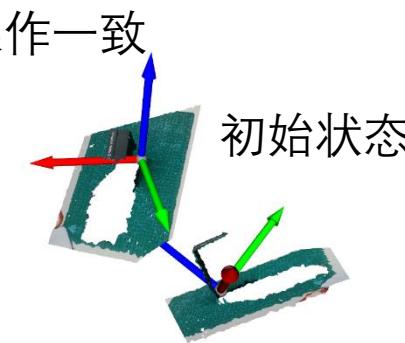
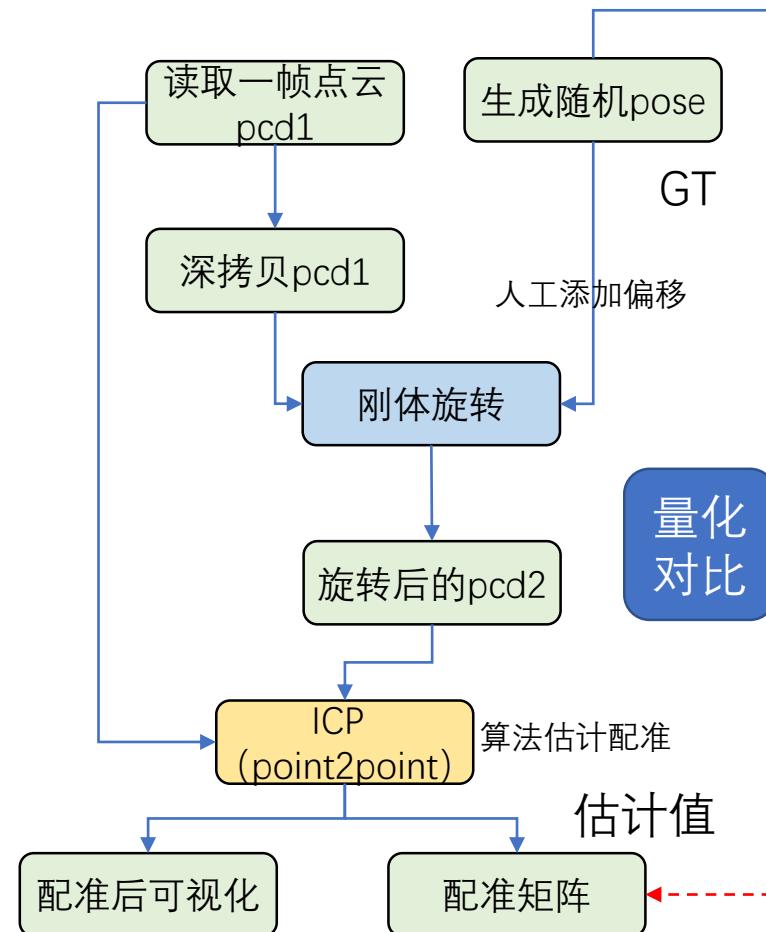
假设有10个三维数据点 = {[4.,7.,0.],[3.,1.,0.],[1.,3.,3.],[5.,4.,6.],[2.,8.,0.],[6.,4.,5.],[1.,1.,8.],[9.,3.,6.],[8.,8.,0.],[0.,1.,8.]}, 计算对应的KD-tree结构。

# Part 5 | Realsense 点云配准

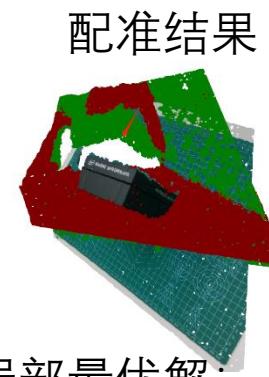
ICP

## » ICP-理想情况

该流程与cloudcompare中的操作一致



初始状态



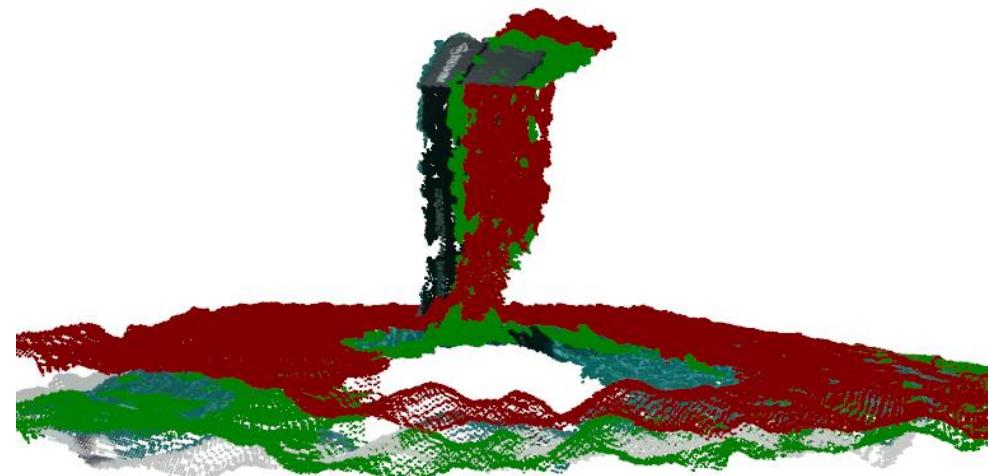
配准结果

初始位置差距较大，陷入局部最优解；  
计算开销较大，对较大规模点云速度较慢。

初始位移小的情况下，能够配准  
一对初始状态依赖强

[[ 0.92475235 -0.07827256 0.37243322 0.09161693]  
 [ 0.19997531 0.93256752 -0.30054565 0.00499302]  
 [-0.32379465 0.35240775 0.87804658 -0.03665762]  
 [ 0. 0. 0. 1. ]]  
 RegistrationResult with fitness=6.243804e-01, inlier\_r=  
 Access transformation to get result.  
 [[ 0.95301973 -0.15621987 -0.25951638 -0.01180124]  
 [ 0.19124756 0.97471397 0.11557269 -0.01125652]  
 [ 0.2348995 -0.15977493 0.95879831 -0.02567311]  
 [ 0. 0. 0. 1. ]]

EST

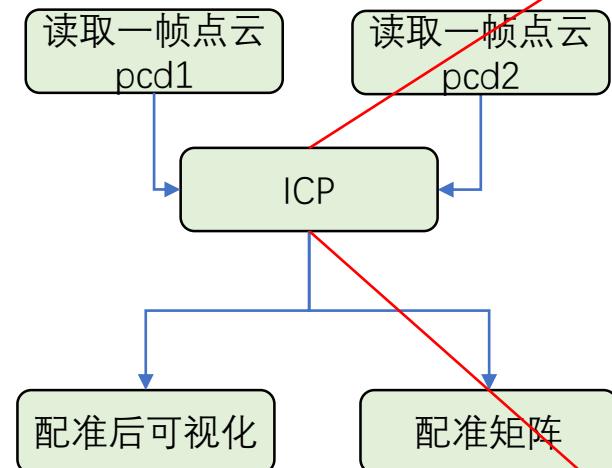


```
# 限制最大转角和最大位移  
max_degree = np.pi/8  
max_dis = 0.1
```

# Part 5 | Realsense 点云配准

ICP

## » ICP-理想情况



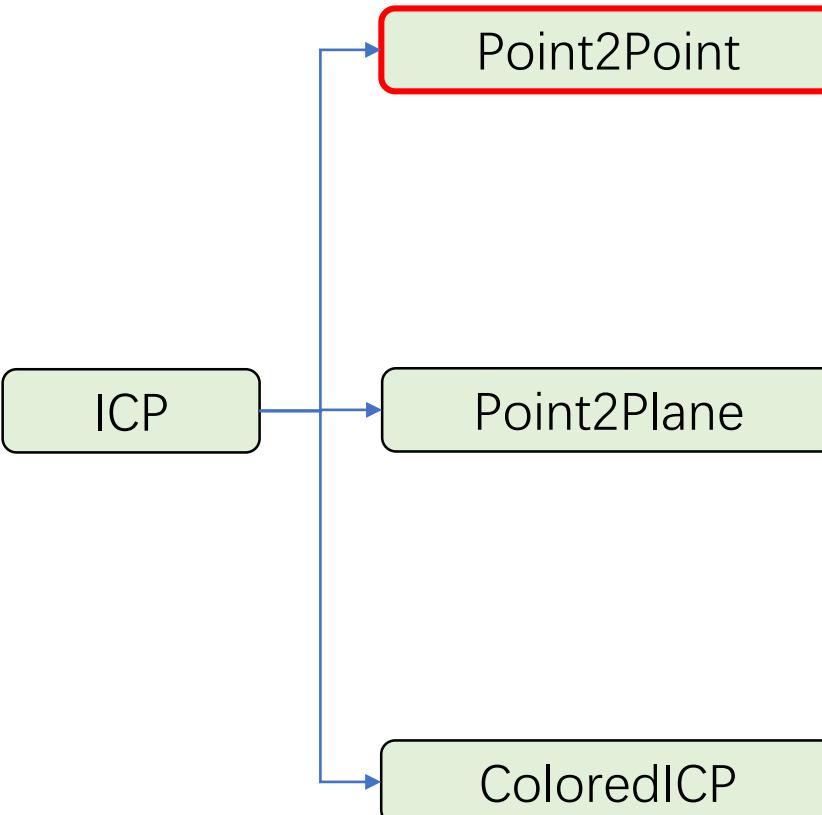
- **初始化:** 假设源点云 (source point cloud) 和目标点云 (target point cloud) , 并设初始变换矩阵为**单位矩阵** (无变换) 。
- **寻找最近点对point-pairs:**
  - 对于源点云中的每个点, 寻找在目标点云中距离最近的点 (最近邻点) , 以此生成对应的点对集合。# 参数即为 threshold
  - 为加速搜索, 使用 KD 树或其他高效的空间索引方法来找到最近邻。
- **估计刚性变换:**
  - 根据找到的点对, 使用最小二乘法估计将源点云对齐到目标点云的刚性变换 (通常包括旋转和平移) 。
  - 计算新的旋转矩阵和平移向量, 使得源点云在新变换下与目标点云尽可能接近。
- **应用变换:**
  - 将计算出的刚性变换应用到源点云上, 以更新源点云的位置。
- **迭代优化:**
  - 重复步骤 2 至 4, 直到满足收敛条件, 例如距离误差小于阈值或达到最大迭代次数。
- **输出结果:**
  - 当满足收敛条件时, 输出最终的旋转矩阵和平移向量, 即实现源点云和目标点云的精确配准。

# Part 5 | Realsense 点云配准

ICP

## » ICP-理想情况

该流程与cloudcompare中的操作一致



1. Find correspondence set  $\mathcal{K} = \{(\mathbf{p}, \mathbf{q})\}$  from target point cloud  $\mathbf{P}$ , and source point cloud  $\mathbf{Q}$  transformed with current transformation matrix  $\mathbf{T}$ .
2. Update the transformation  $\mathbf{T}$  by minimizing an objective function  $E(\mathbf{T})$  defined over the correspondence set  $\mathcal{K}$ .

目标函数

$$E(\mathbf{T}) = \sum_{(\mathbf{p}, \mathbf{q}) \in \mathcal{K}} \|\mathbf{p} - \mathbf{T}\mathbf{q}\|^2$$

即通过迭代优化寻找  
一组 $\mathbf{T} = [R, t]$ 矩阵使得 $\mathbf{p}$ 、  
 $\mathbf{q}$ 的欧式距离最短。

```
threshold = 0.02
trans_init = np.eye(4)
reg_p2p =
o3d.pipelines.registration.registration_icp(
    target, source,
    threshold,
    trans_init,
    o3d.pipelines.registration.TransformationEsti-
    mationPointToPoint())
```

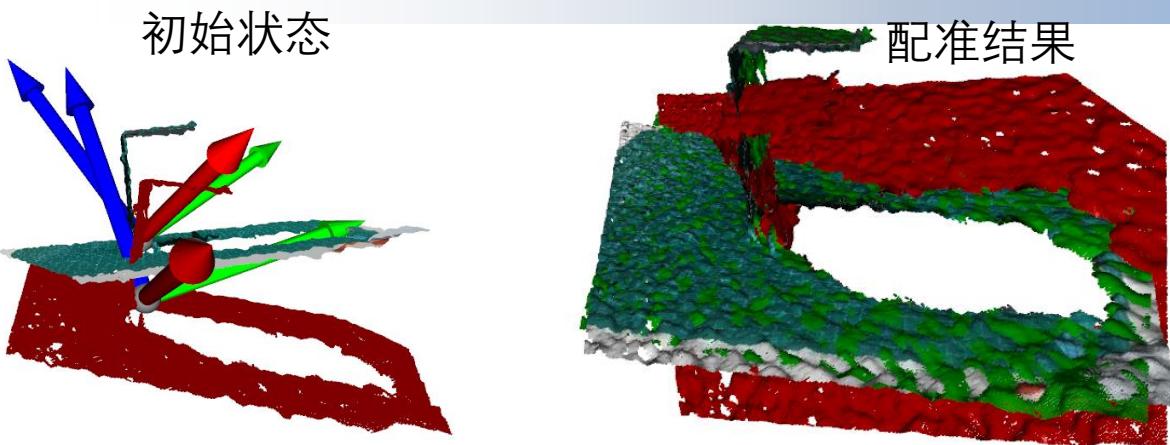
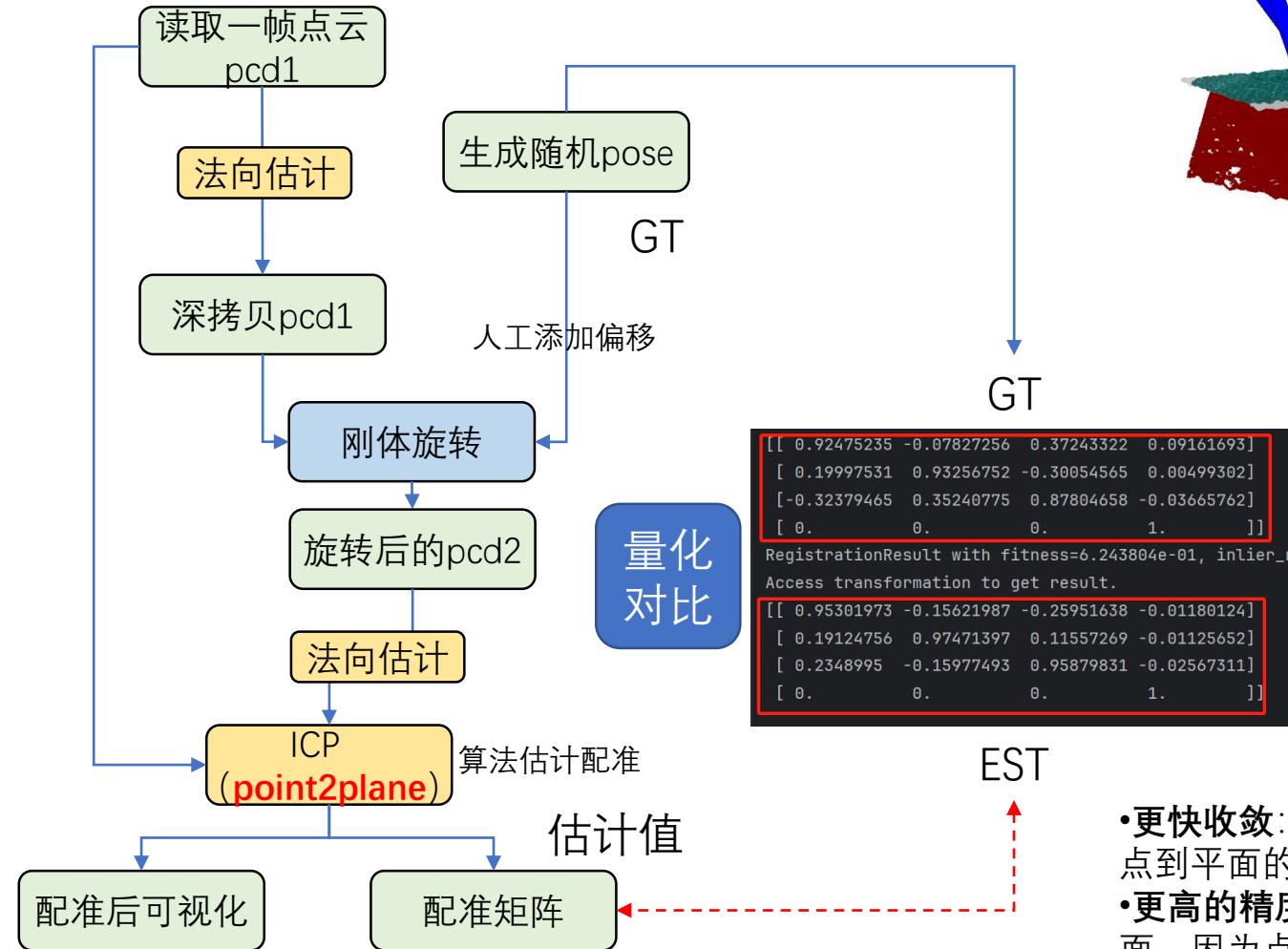
- ①②点云target, 点云source (固定)
- ③ 配准搜索(使用KD-Tree)距离,  
太小搜不到特征点, 太大引入误匹配
- ④粗匹配, 可通过全局配准获得,  
一般为单位矩阵
- ⑤ 配准算法

# Part 5 | Realsense 点云配准

ICP

该流程与cloudcompare中的操作一致

## ICP-理想情况



Point-to-Plane: plane相当于一簇点，在没有类似滑动窗口的情况下，类似于寻找一组具有共性特征的点簇参与计算。

可以尝试增加随机化范围( $R, t$ )  
单独增大，缩小旋转，或者平移，或者控制自在  
一个自由度进行平移，观测算法的执行结果。

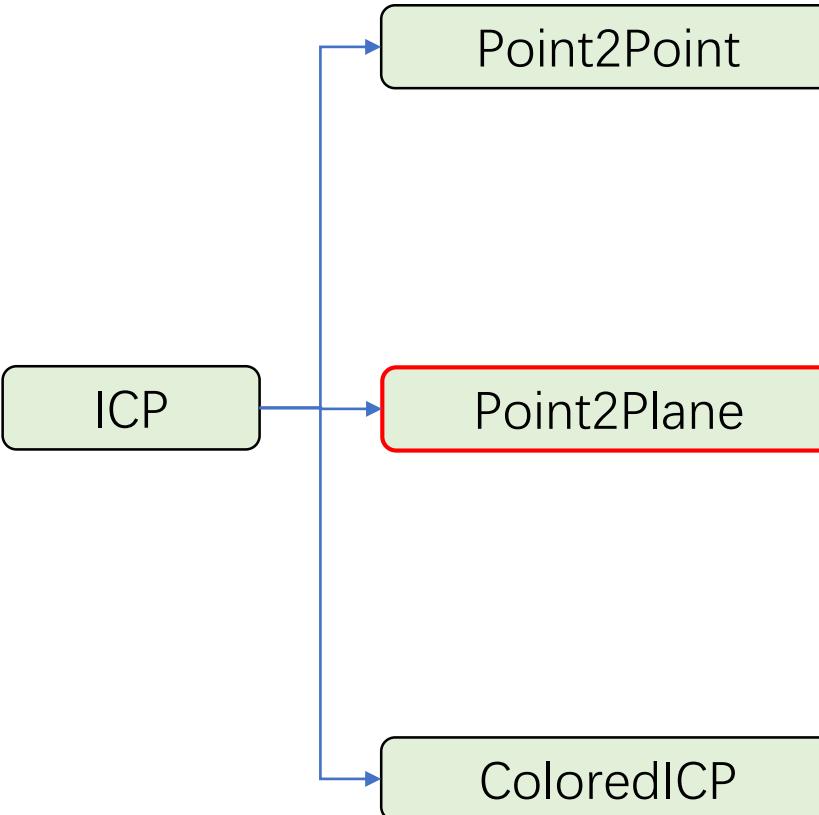
- 更快收敛**: Point-to-Plane ICP 通常比 Point-to-Point ICP 收敛更快，因为点到平面的距离可以更加有效地减少误差，使优化过程更有效率。
  - 更高的精度**: Point-to-Plane ICP 能够更精确地对齐具有复杂几何形状的表面，因为点到平面的距离更符合表面间的几何关系，减少配准误差。
  - 鲁棒性**: 由于减少了目标函数中的自由度，Point-to-Plane ICP 对初始配准误差的容忍度更高，适用于复杂场景中的配准问题。

# Part 5 | Realsense 点云配准

ICP

## » ICP-理想情况

该流程与cloudcompare中的操作一致



1. Find correspondence set  $\mathcal{K} = \{(\mathbf{p}, \mathbf{q})\}$  from target point cloud  $\mathbf{P}$ , and source point cloud  $\mathbf{Q}$  transformed with current transformation matrix  $\mathbf{T}$ .
2. Update the transformation  $\mathbf{T}$  by minimizing an objective function  $E(\mathbf{T})$  defined over the correspondence set  $\mathcal{K}$ .

目标函数

$$E(\mathbf{T}) = \sum_{(\mathbf{p}, \mathbf{q}) \in \mathcal{K}} ((\mathbf{p} - \mathbf{T}\mathbf{q}) \cdot \mathbf{n}_p)^2,$$

即通过迭代优化寻找  
一组 $\mathbf{T}=[R, t]$ 矩阵使得 $\mathbf{q}$   
到 $\mathbf{p}$ 的平面点（ $\mathbf{n}_p$ 为法  
向）的欧式距离最短。

此外，还依赖法向信息，所以需要在计算前，先估计source的法向。

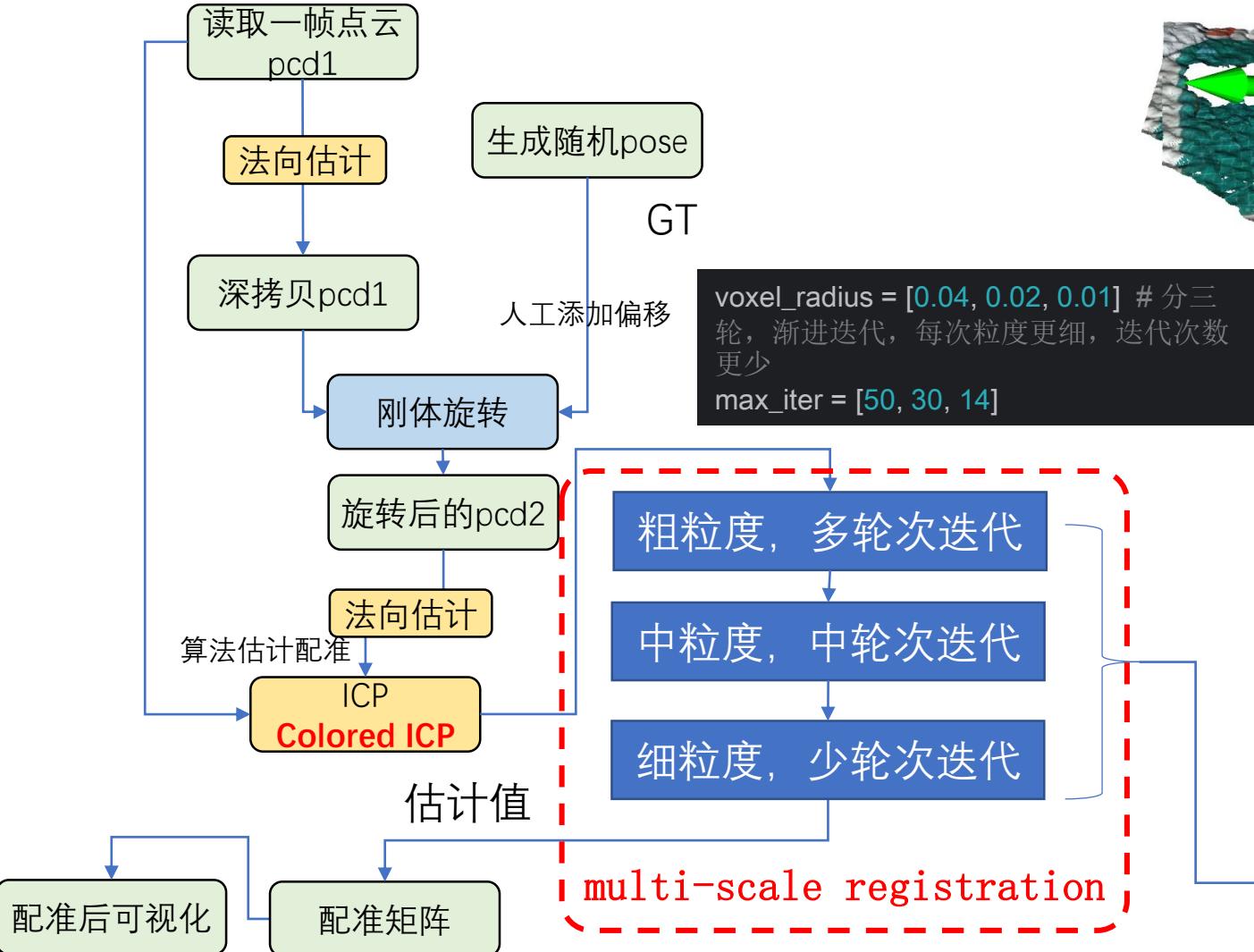
```
pcd2.estimate_normals(search_param=o3d.geometry.KDTreeSearchParamHybrid(radius=0.1, max_nn=30))
```

因此，我们可以断定，当传感器的运动采样间隔导致 $\mathbf{T}$ 的变化过大时，采用上述算法，将很难获得理想配准结果！！！

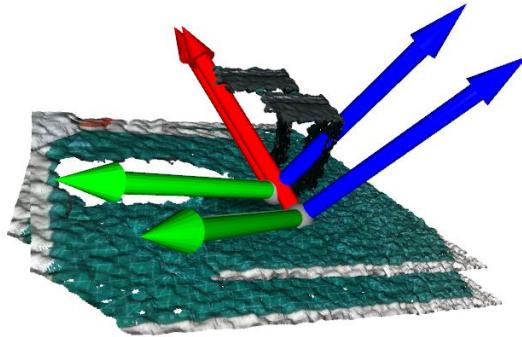
# Part 5 | Realsense 点云配准

ICP

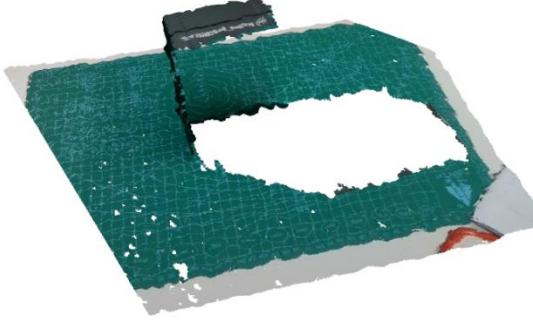
## » ICP-理想情况



初始状态



配准结果



Point-to-Plane: plane相当于一簇点，在没有类似滑动窗口的情况下，类似于寻找一组具有共性特征的点簇参与计算。

可以尝试增加随机化范围(R,t)  
单独增大, 缩小旋转, 或者平移, 或者控制自在一个自由度进行平移, 观测算法的执行结果。

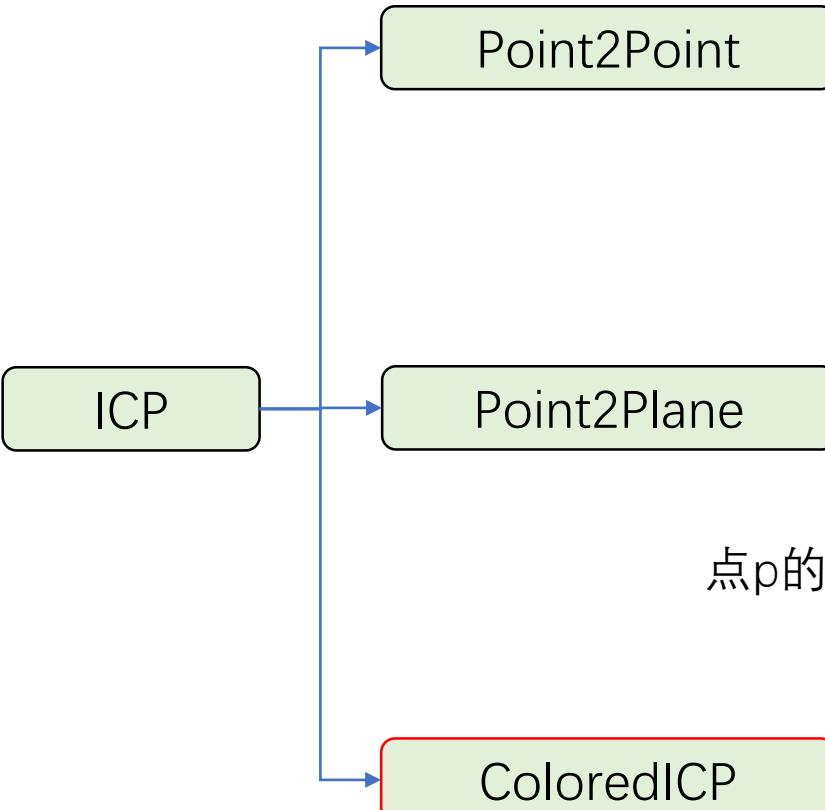
注: 此算法也并非万能, 一旦中间迭代配准找不到特征点算法还是会崩溃

# Part 5 | Realsense 点云配准

ICP

## » ICP-理想情况

该流程与cloudcompare中的操作一致



1. Find correspondence set  $\mathcal{K} = \{(\mathbf{p}, \mathbf{q})\}$  from target point cloud  $\mathbf{P}$ , and source point cloud  $\mathbf{Q}$  transformed with current transformation matrix  $\mathbf{T}$ .
2. Update the transformation  $\mathbf{T}$  by minimizing an objective function  $E(\mathbf{T})$  defined over the correspondence set  $\mathcal{K}$ .

目标函数

$$E(\mathbf{T}) = (1 - \delta)E_C(\mathbf{T}) + \delta E_G(\mathbf{T})$$

色彩部分

几何部分: point2plane

$$E_C(\mathbf{T}) = \sum_{(\mathbf{p}, \mathbf{q}) \in \mathcal{K}} (C_p(f(\mathbf{T}\mathbf{q})) - C_q(\mathbf{q}))^2, \quad E_G(\mathbf{T}) = \sum_{(\mathbf{p}, \mathbf{q}) \in \mathcal{K}} ((\mathbf{p} - \mathbf{T}\mathbf{q}) \cdot \mathbf{n}_p)^2,$$

点 $p$ 的颜色 $C(p)$  到点对应的切平面投影的点 $C_p(f(Tq))$ 颜色 $C(q)$ 的差别

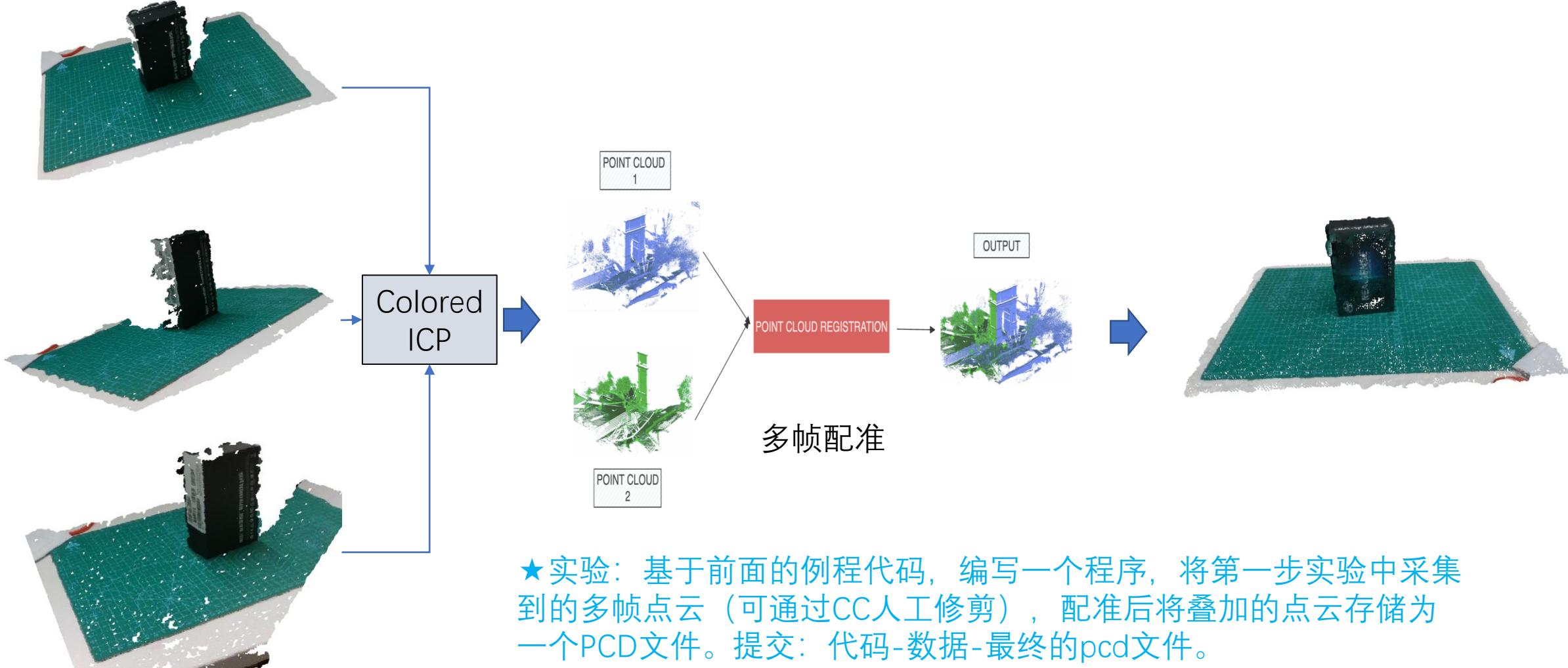
★实验: 从采集数据中的一帧点云, 添加随机旋转平移, 然后重新配准: 使用ICP point2point, point2plane, colored, 并调节参数, 观察结果。

注意关闭着色

# Part 5 | Realsense 点云配准

ICP

## » ICP-连续帧处理



★实验：基于前面的例程代码，编写一个程序，将第一步实验中采集到的多帧点云（可通过CC人工修剪），配准后将叠加的点云存储为一个PCD文件。提交：代码-数据-最终的pcd文件。

# 开始实验



北京航空航天大学  
BEIHANG UNIVERSITY

# Part 5 Realsense传感器初探

# 传感器介绍

## 》 传感器数据获取

Velodyne激光雷达参数对比（公开数据集可能用到，已退出大陆市场）

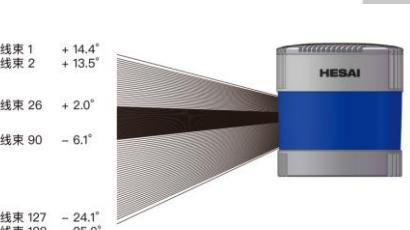


图 1.5 线束分布示意图

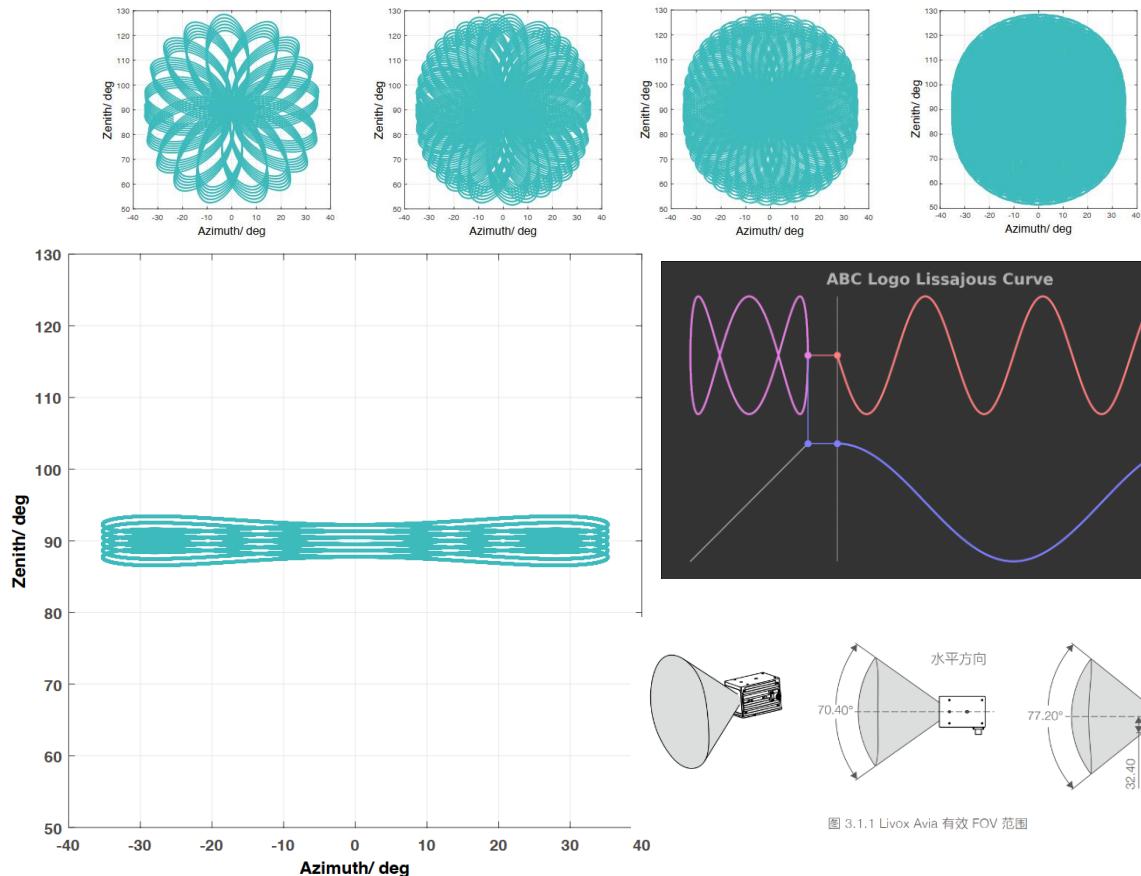
# Part 5 | Realsense传感器初探

## 传感器介绍

### 》 传感器数据获取

非重复扫描激光雷达Livox参数对比。

包含两种扫描模式：玫瑰线rosedosventor、李萨如（Lissajous）



产品型号	MID-40	HAP	TELE-15	MID-70	AVIA
应用场景	移动机器人、智慧物流等	自动驾驶、智能辅助驾驶系统等	轨道交通、智慧交通	无人配送车、园区物流车等	测绘、安防、智慧城市等
量程 <sup>1</sup> (@ 100 klx)	90 m @ 10% 反射率 130 m @ 20% 反射率 260 m @ 80% 反射率	150 m @ 10% 反射率 200 m @ 80% 反射率	320 m @ 10% 反射率 500 m @ 50% 反射率	90 m @ 10% 反射率 130 m @ 20% 反射率 260 m @ 80% 反射率	190 m @ 10% 反射率 230 m @ 20% 反射率 320 m @ 80% 反射率 450 m @ 80% 反射率, 0 klx
虚警率 <sup>2</sup> (@ 100 klx)	< 0.01%	< 0.01%	< 0.01%	< 0.01%	< 0.0003%
距离精度 <sup>3</sup> (1σ @ 20m)	2 cm	2 cm	2 cm	2 cm	2 cm
角度精度	< 0.05°	< 0.1°	< 0.03°	< 0.1°	< 0.05°
水平视场 (FOV)	38.4°	120°	14.5°	70.4°	70.4°
垂直视场 (FOV)	38.4°	25°	16.1°	70.4°	77.2°
光束发散度	0.28° (垂直) × 0.03° (水平)	0.28° × 0.03°	0.12° × 0.02°	0.28° × 0.03°	0.28° × 0.03°
数据率 (点/秒)	~100,000	452,000	~240,000 ~480,000(双回波)	~100,000 ~200,000(双回波)	~240,000 ~480,000(双回波) ~720,000(三回波)
供电电压范围	10 ~ 16 V	9 ~ 18 V	10 ~ 15 V	10 ~ 15 V	10 ~ 15 V
功率	10 W(启动功率 25W)	14 W(启动功率 26W)	12 W(启动功率 30W)	8 W(启动功率 30W)	9 W(启动功率 16W)
尺寸 (mm)	88×69×76	105×131.6×65	122×105×95 无风扇时 112×122×85	97×64×62.7	91×61.2×64.8
重量	760 g	1120g	1600g 无风扇时 1500g	580g	498g
工作温度	-20°C to 65°C	-40°C to 85°C	-40°C to 85°C	-20°C to 65°C	-20°C to 65°C
数据同步	IEEE 1588-2008 (PTPv2) PPS (Pulse Per Second)	IEEE 802.1AS (gPTP)	IEEE 1588-2008 (PTPv2) PPS (Pulse Per Second)	IEEE 1588-2008 (PTPv2) PPS (Pulse Per Second)	IEEE 1588-2008 (PTPv2) PPS (Pulse Per Second)
接口	Ethernet	HAP (T1) : 100 Base-T1 标准 HAP (TX) : 100 Base-TX 标准	Ethernet	Ethernet	Ethernet
激光波长及安全级别	905 nm Class 1 人眼安全	905 nm Class 1 人眼安全	905 nm Class 1 人眼安全	905 nm Class 1 人眼安全	905 nm Class 1 人眼安全