



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)

## Lecture4-点云语义标注&3D分割模型

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 助教：张宇辰



邮箱：ouyangkid@buaa.edu.cn



学期：2024年秋季

# 目录

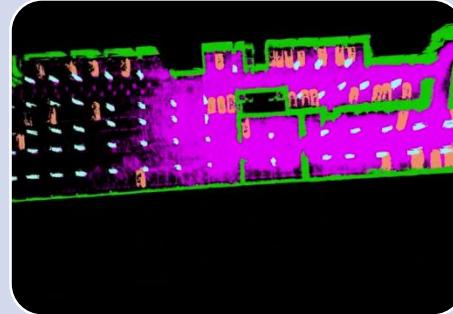
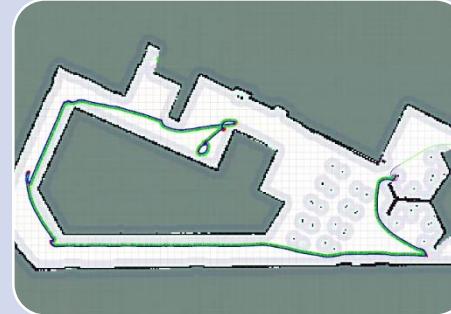
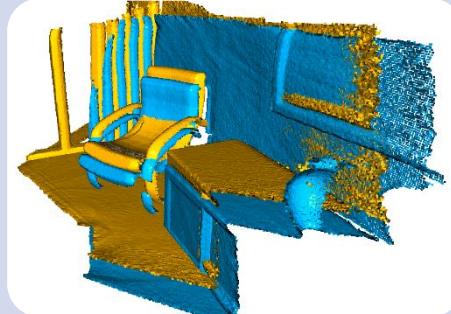
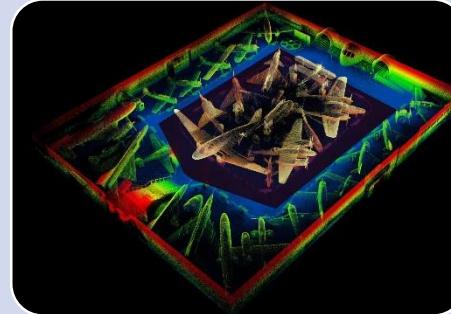
## Contents

- 01 课程内容
- 02 课程目标
- 03 学习目标
- 04 实验设备及数据
- 05 点云语义标注&3D分割模型

# 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 点云语义标注&3D分割模型

# 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強。

!

- 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 点云语义标注&3D分割模型

# 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3作业（以子123文件夹的形式组织）：
  - 基于bag文件和pose.txt进行LOAM算法配置，通过回放bag生成算法估计的位姿结果，存为pose\_loam.txt。同时，利用数据存储节点，将执行过程的点云存为pcd，将位姿存为odom文件。
  - 使用EVO工具对比给出的pose.txt和本地算法结果pose\_loam.txt，绘制pose轨迹，APE和RPE。
  - 通过interactive\_slam工具，导入pcd+odom，进行手动地图关键帧选取和地图优化，保存优化后的点云地图为pcd。
  - 提交上述程序中生成的文件，以及对比的可视化结果（红色内容）。

# 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）

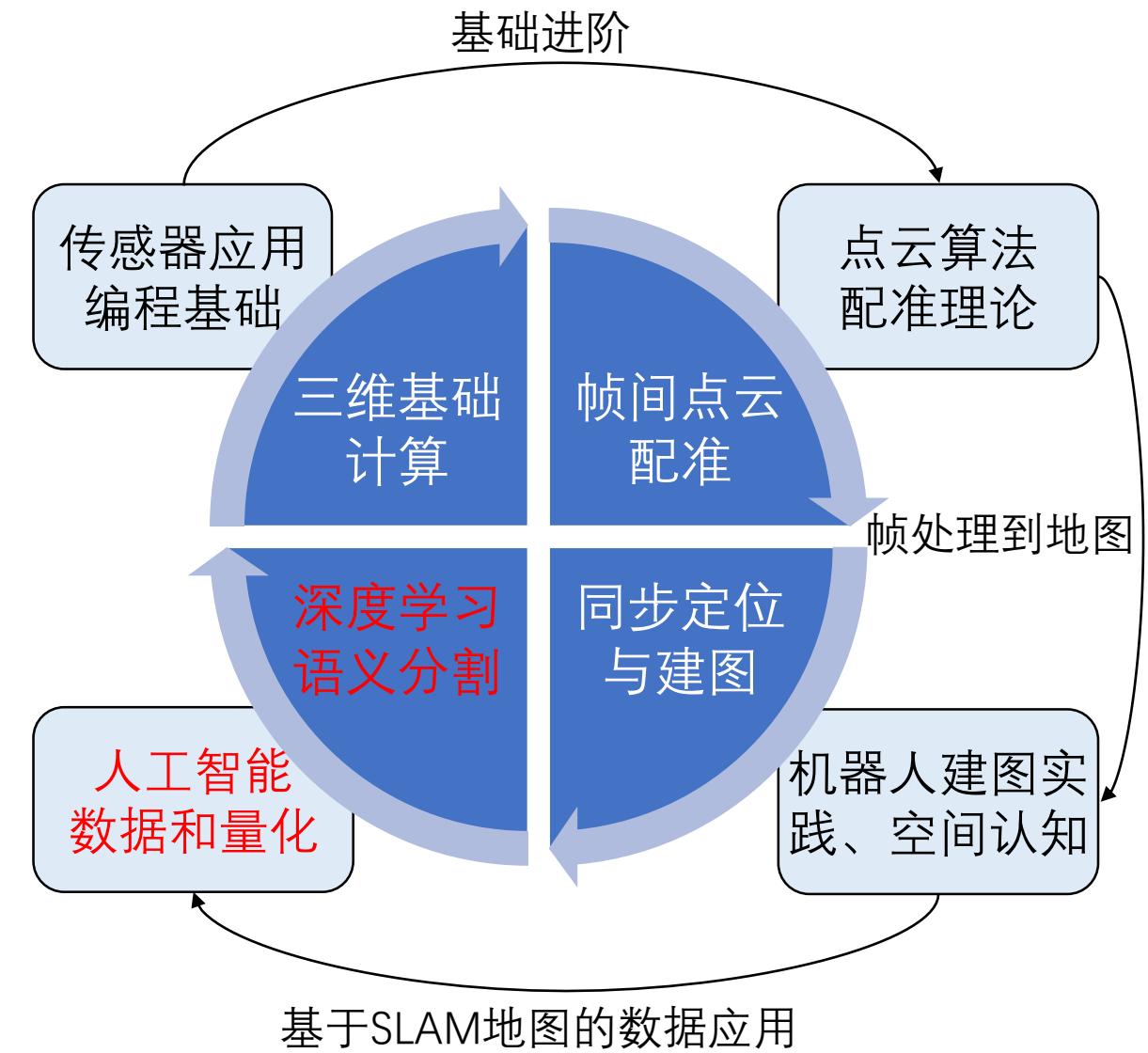
- Realsense：双目+结构光，稠密视场点云
- LiDAR：机械ToF测距，稀疏环视点云

### 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）

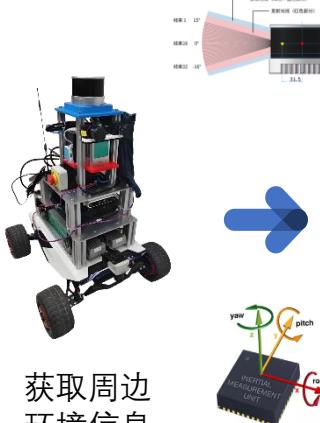
- Ubuntu: shell
- ROS机器人操作系统
- PointLabeler



# Part 3 | 学习目标

## 3-4课整体架构

### » 3D点云数据处理和可视化实践入门

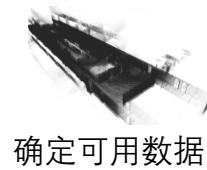
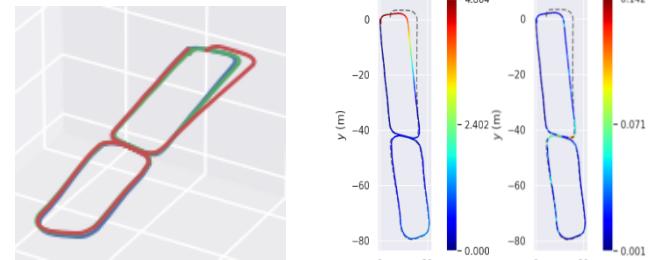
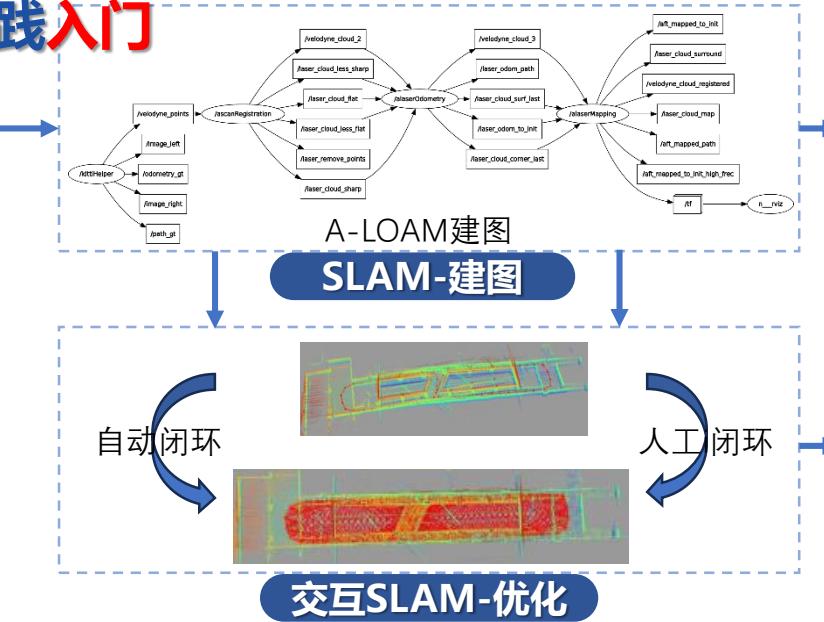


获取周边环境信息



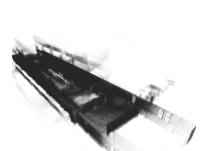
bag数据保存于NAS

数据准备

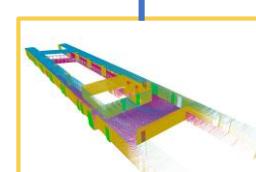


确定可用数据

评估



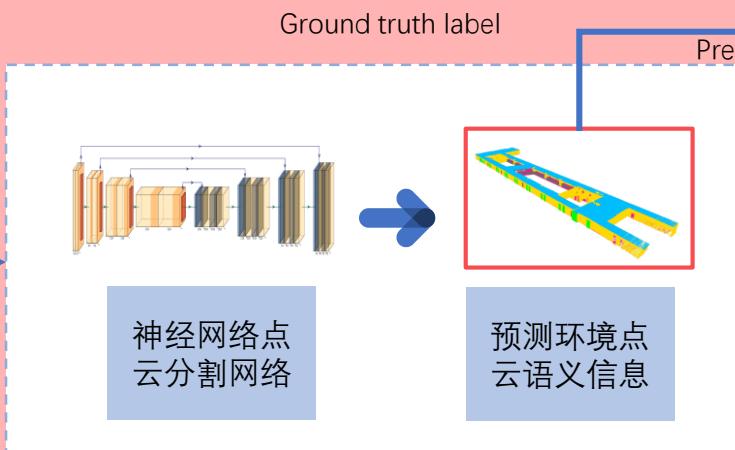
接收kitti  
格式数据



Point  
Labeler  
数据标注

赋予语义  
标签

数据准备



$$IoU = \frac{TP}{TP + FN + FP}$$
$$MIoU = \frac{1}{k+1} \sum_{i=0}^k \frac{TP}{FN + FP + TP}$$

各label种类的分割精度

训练-推理

评估

# 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 点云语义标注&3D分割模型

# Part 4 | NAS涉及数据

## » 3D点云数据处理和可视化实践

<input type="checkbox"/>	名称	修改日期	类别	大小	
<input type="checkbox"/>	Code	2024/11/12 07:25:40	文件夹		
<input type="checkbox"/>	dataset	2024/11/12 06:47:59	文件夹		
<input type="checkbox"/>	point_labeler.wiki	2024/11/12 02:35:48	文件夹		
<input type="checkbox"/>	software	2024/11/08 07:59:14	文件夹		
<input type="checkbox"/>	R1kitti32	2024/11/07 14:39:05	文件夹		
<input type="checkbox"/>	checkpoint_epoch_36.pth	2024/11/08 12:20:26	PTH 文件	113.7 MB	
<input type="checkbox"/>	labels.xml	2024/10/31 07:14:20	XML 文件	2.24 KB	

# 目录

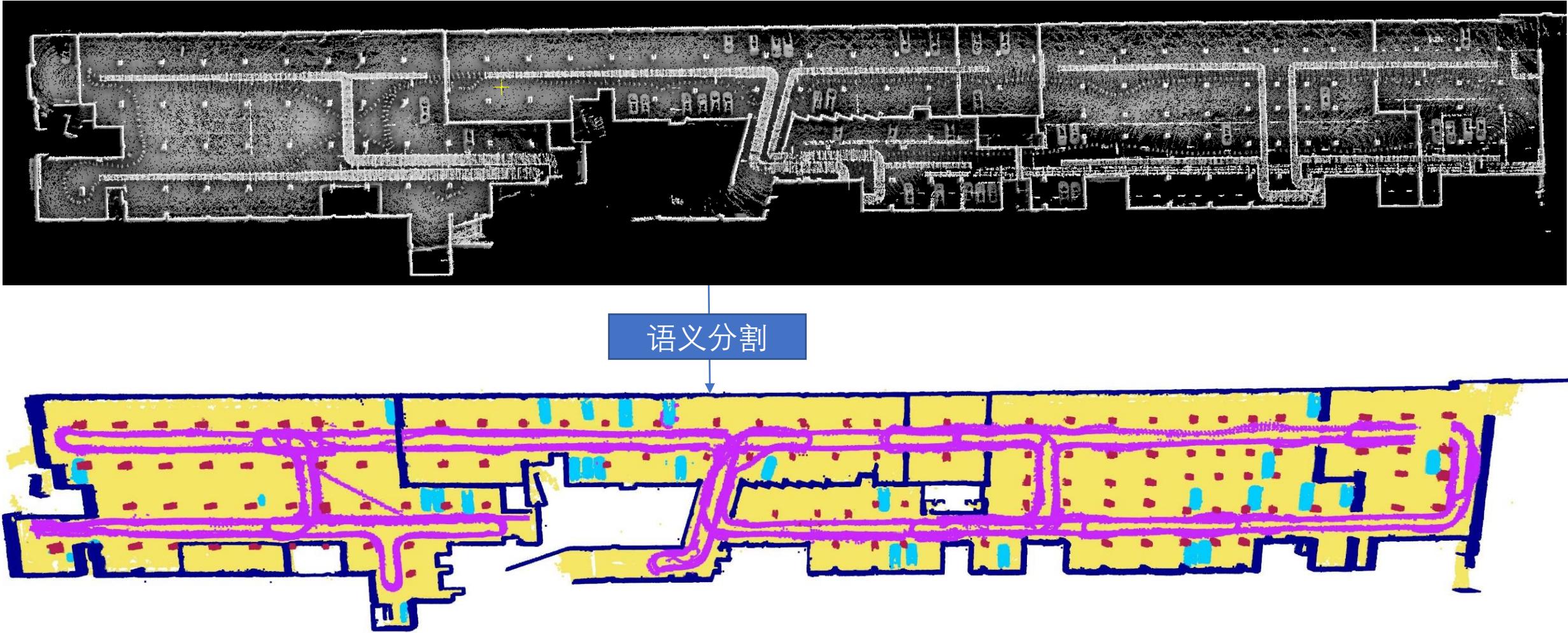
## Contents

- 01 点云配准
- 02 课程目标
- 03 学习目标
- 04 实验设备及数据
- 05 点云语义标注&3D分割模型

# Part 5 | 激光SLAM

SLAM简介

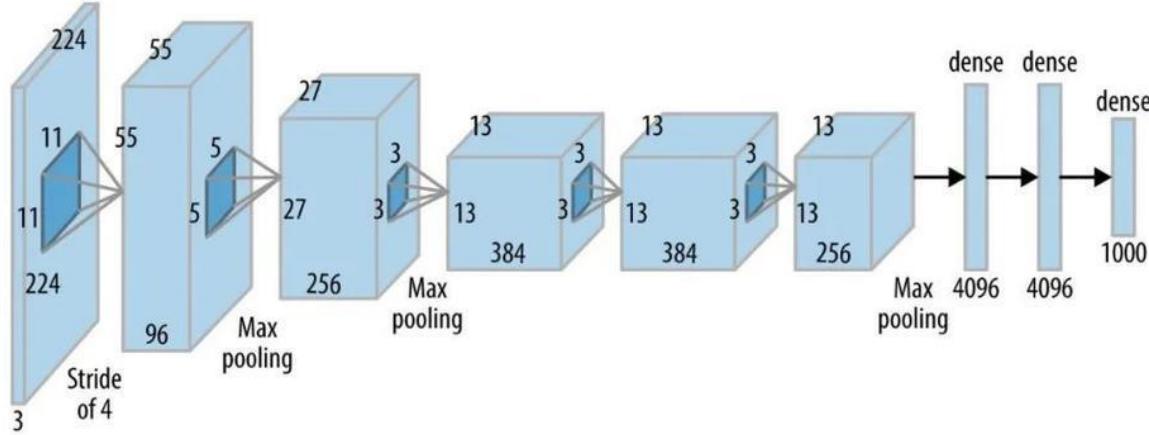
## » 激光雷达



# Part 5 | 卷积

2D卷积

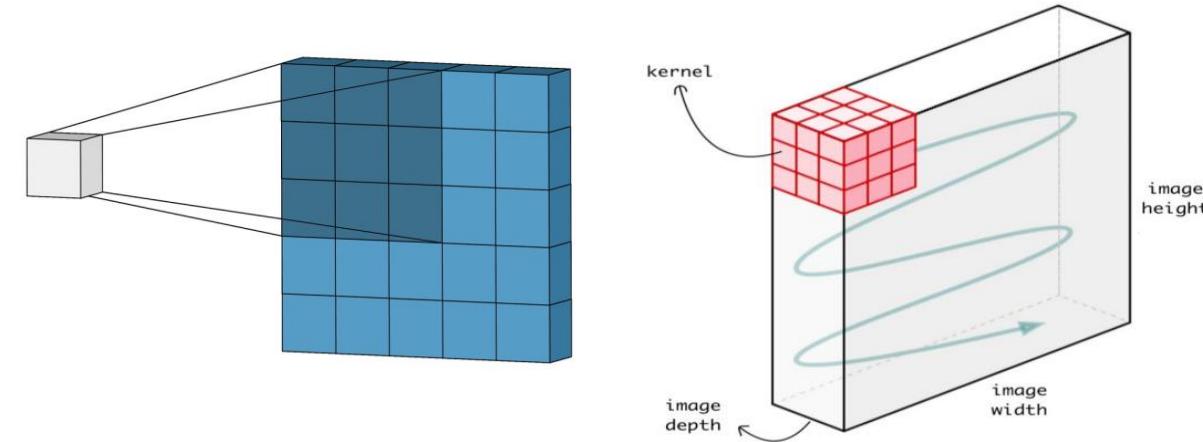
## » 2D vs. 3D



ALEXNET

深度学习感知首先在图像领域取得进展，神经网络卷积计算主要面向2D，因为输入图像平面是一个多通道矩阵，借助已有的图像处理经验—使用像素及其邻域的拓扑特征，而非单一像素—卷积核。

卷积计算模拟人类感受野，由粗到精的抽取图像特征。结合随机梯度下降搜索，以及误差反向传播更新网络参数。从而实现各种感知任务



# Part 5 | 卷积

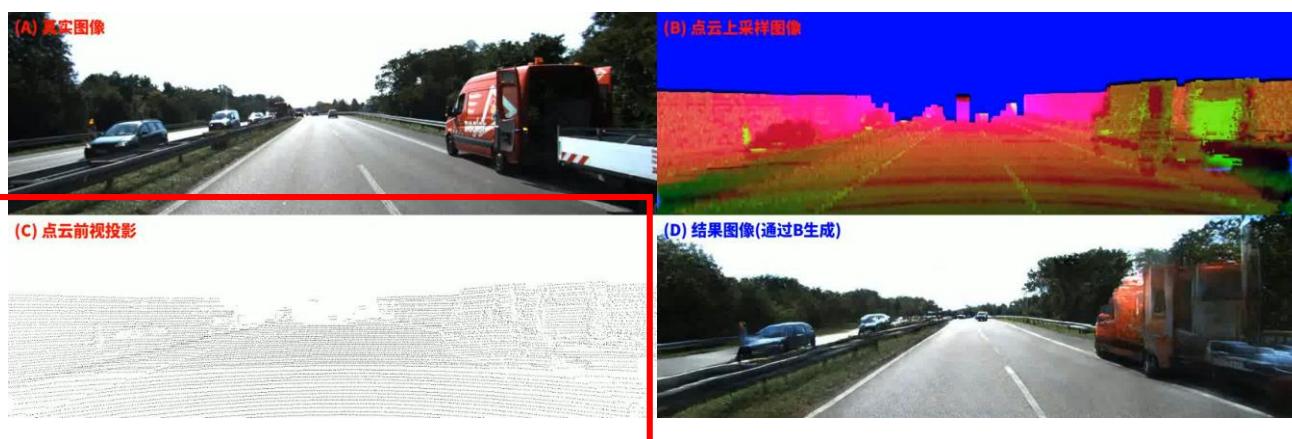
## » 2D vs. 3D

- Rangemap-based策略

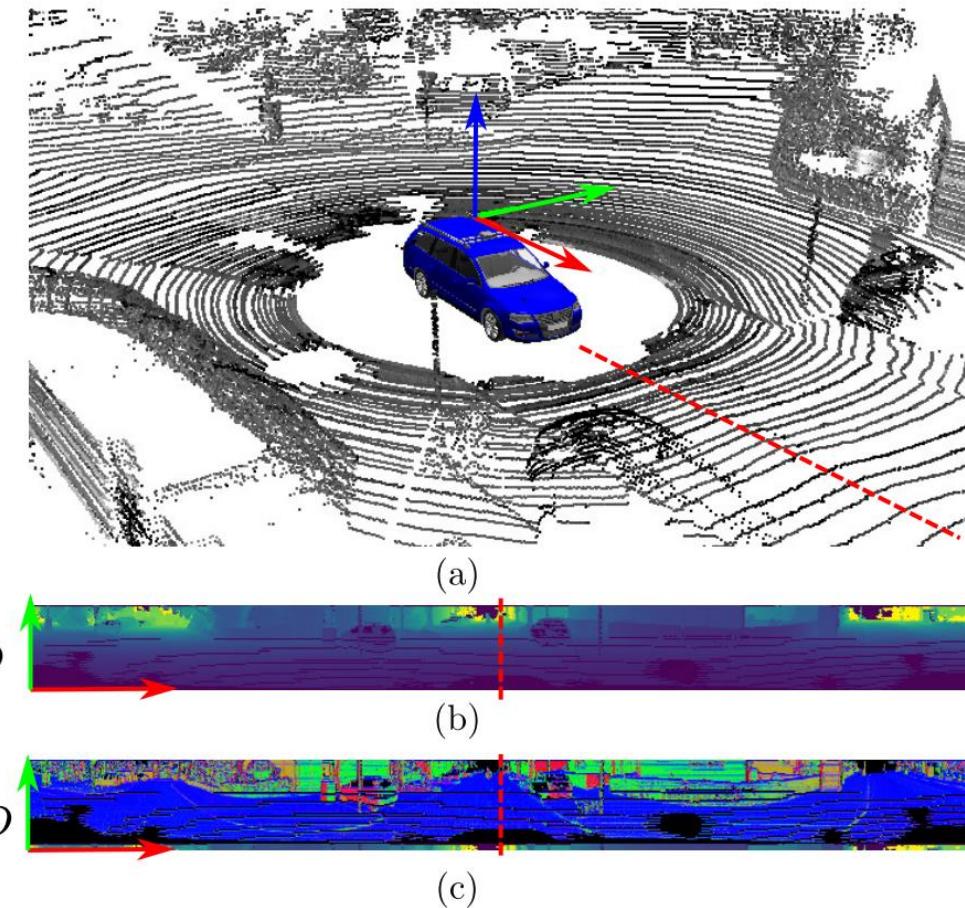
Rangemap基于雷达模型的球坐标系进行线投影，以每个点为像素进行平面映射，将三维离散点云降维到特征稠密2D伪视图。

然后借助图像处理的深度学习模型进行求解—**速度快、精度低**。

**缺点：**丢失维度信息，损失空间关联，投影失真，投影和逆投影还会丢失一些重叠点。



$\mathcal{P}$



pixel contains the nearest 3D point. Each point  $\mathbf{p}_i = (x, y, z)$  is converted via the function  $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$  to spherical coordinates and finally to image coordinates  $(u, v)$ , i.e.,

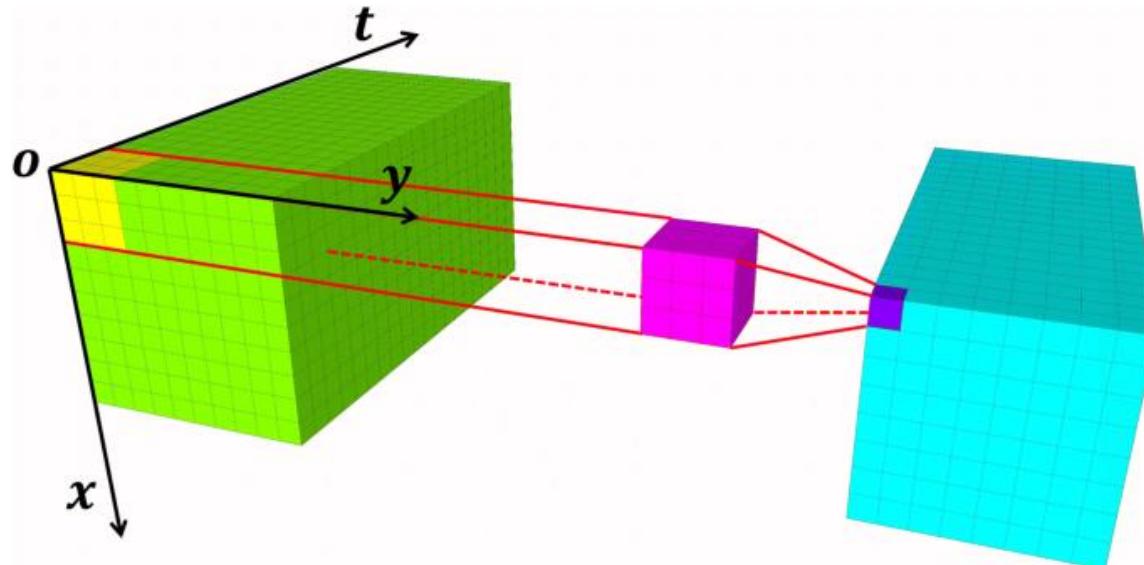
$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2} [1 - \arctan(y, x) \cdot \pi^{-1}] \cdot w \\ [1 - (\arcsin(z \cdot r^{-1}) + f_{\text{up}}) f^{-1}] \cdot h \end{pmatrix}, \quad (2)$$

where  $r = \|\mathbf{p}\|_2$  is the range,  $f = f_{\text{up}} + f_{\text{down}}$  is the vertical field-of-view of the sensor, and  $w, h$  are the width and height of the resulting vertex map  $\mathcal{V}_D$ . This projection function is

# Part 5 | 卷积

## » 2D vs. 3D

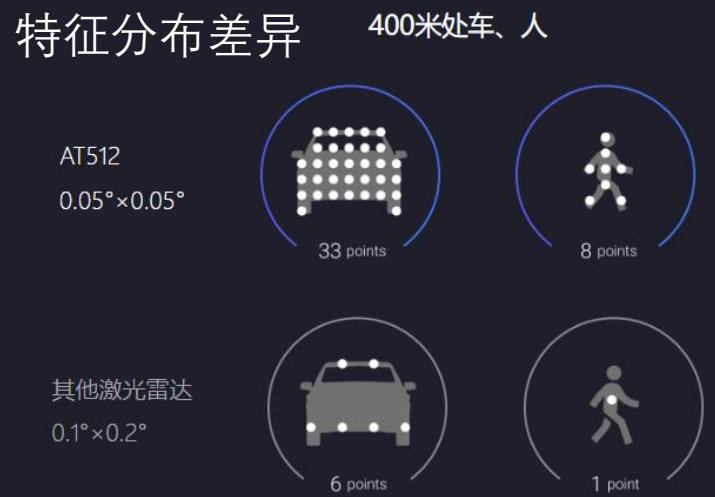
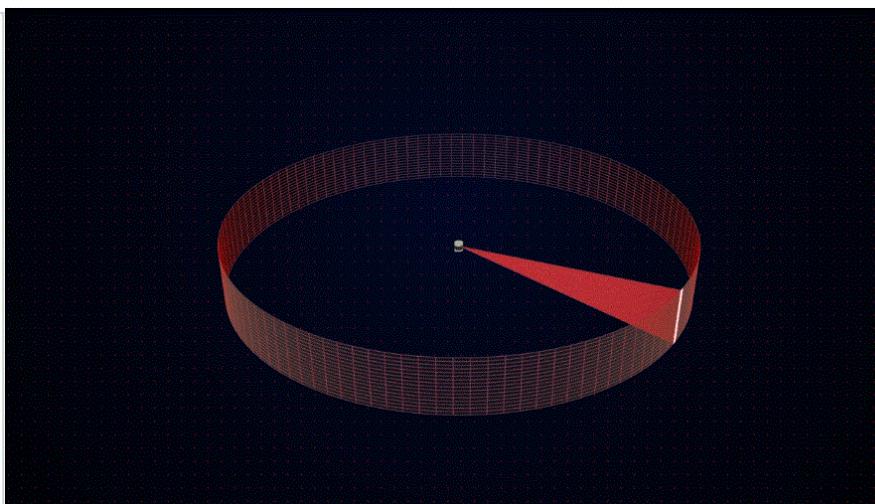
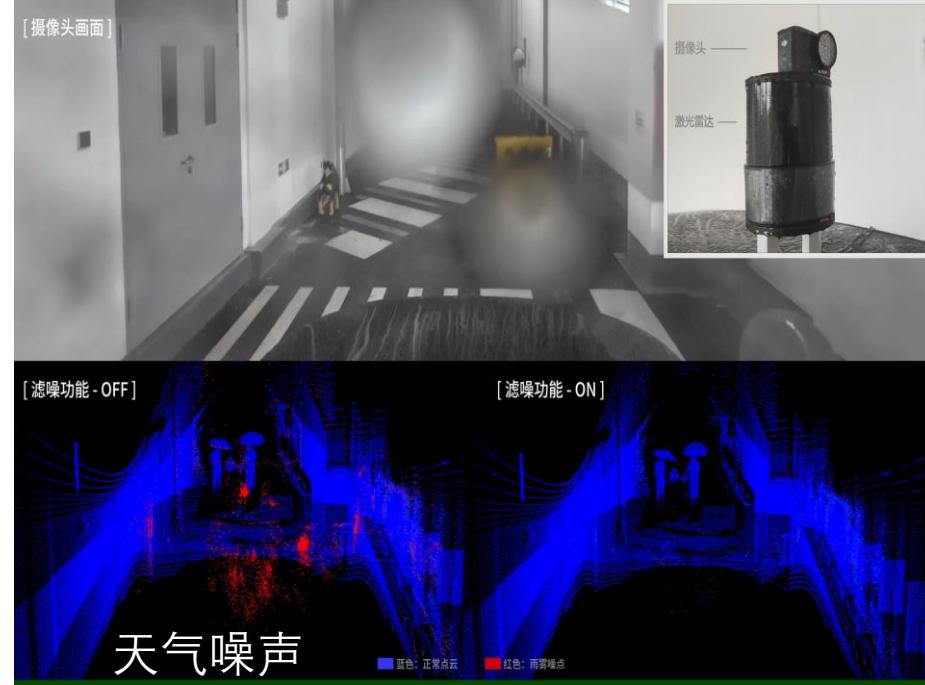
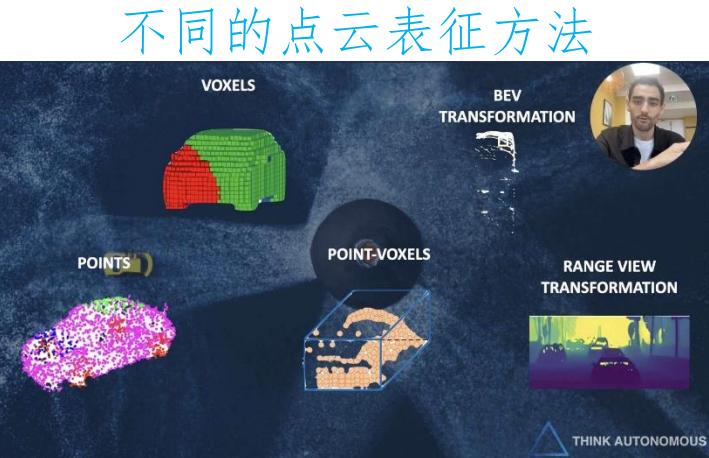
- 当处理图像序列任务时，如视频流、CT断层扫描，具有相同数据结构的图像随着观测时间积累，增加了 $t$ 维度。
- 同时，仅限于（单帧）空间维度的特征无法支持更加复杂的感知任务时，需要对三维数据进行卷积特征提取，如追踪、预测、Re-ID、SLAM。
- 3D卷积在2D的基础上，对卷积核维度进行扩展，即可获取三维空间特征提取和编码的能力。



# Part 5 | 卷积

## » 2D vs. 3D

- 三维点云数据特性：
  - (垂直) 视场角有限
  - 随机、无序、缺失未知(超感知范围能量消散)
  - 视距内ToF扫描，仅能够感知目标**表面结构**
  - 激光的射散：多次回波会形成更多的返回点
  - 透明目标和非朗博体形成的测量噪声
  - 天气噪声干扰（强光、雾气、雨水、沙尘）



# Part 5 | 卷积

## » 2D vs. 3D

### • 三维特征工程（2016以前）：

**Abstract** A number of 3D local feature descriptors have been proposed in the literature. It is however, unclear which descriptors are more appropriate for a particular application.

A good descriptor should be descriptive, compact, and robust to a set of nuisances. This paper compares ten popular local

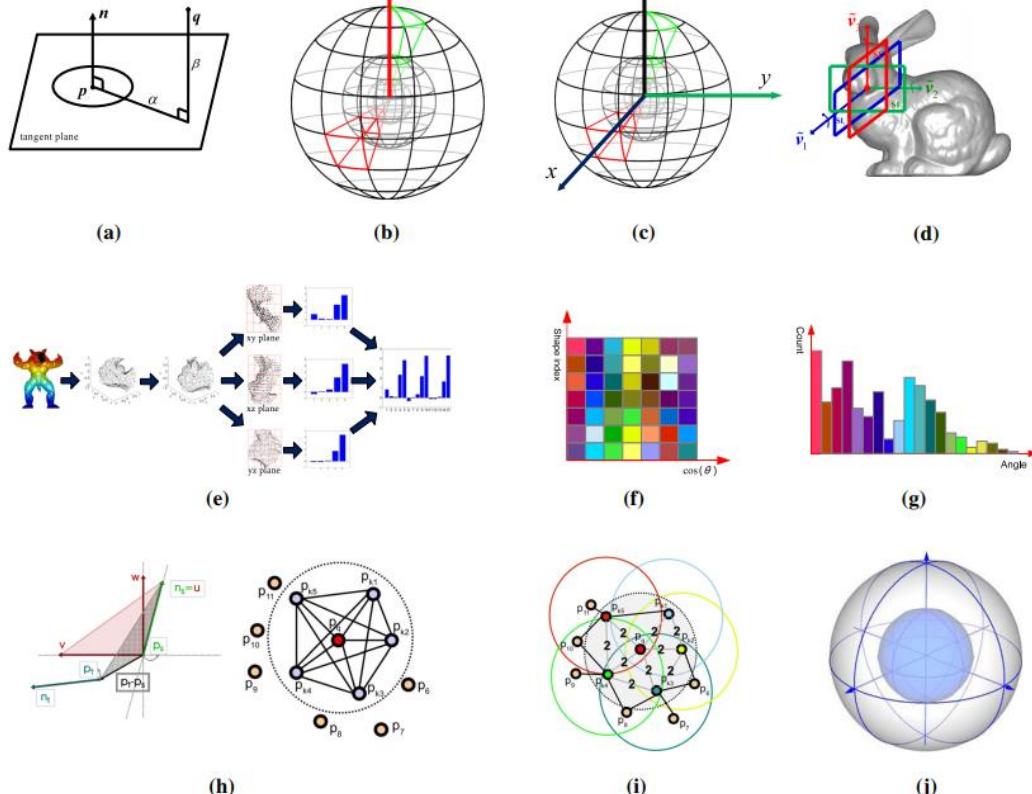
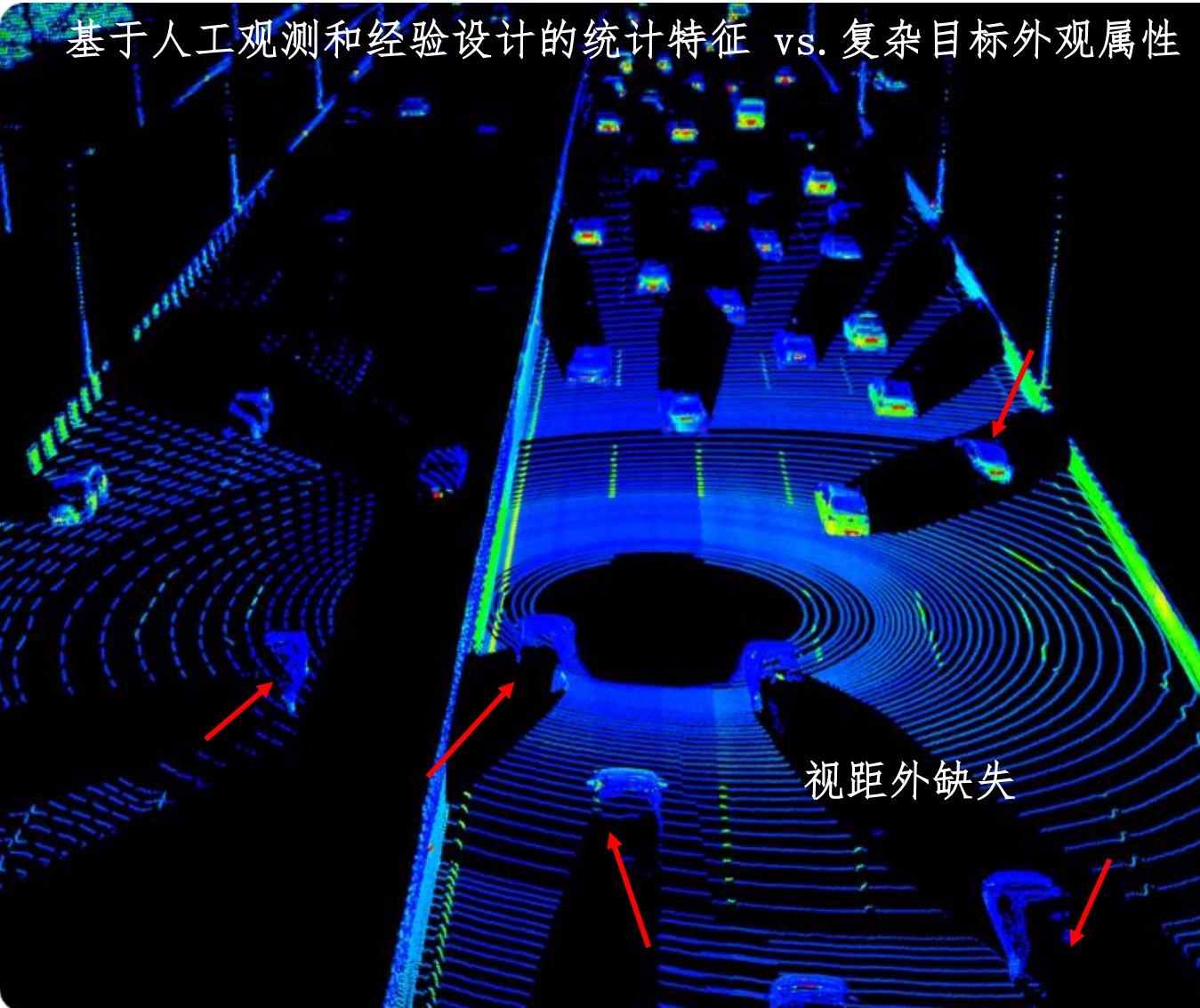


Fig. 1 A schematic illustration of the selected descriptors. a SI b 3DSC c USC d TriSIFT e RoPS f LSP g THRIFT h PFH i FPFH j SHOT (Color figure online)

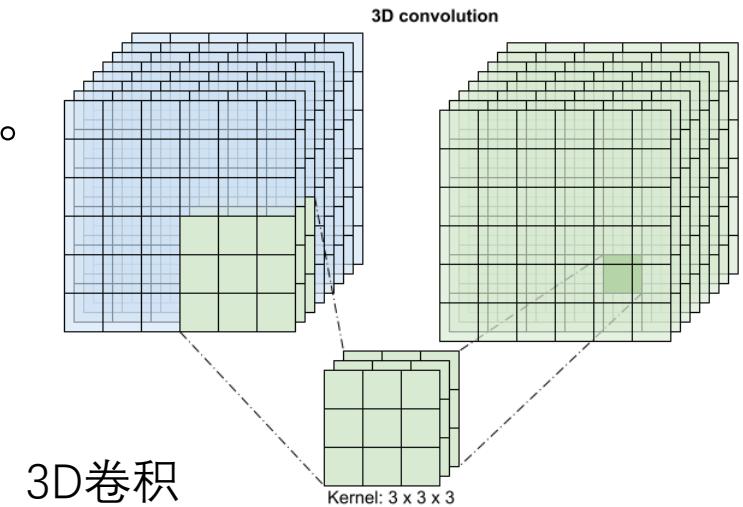


# Part 5 | 卷积

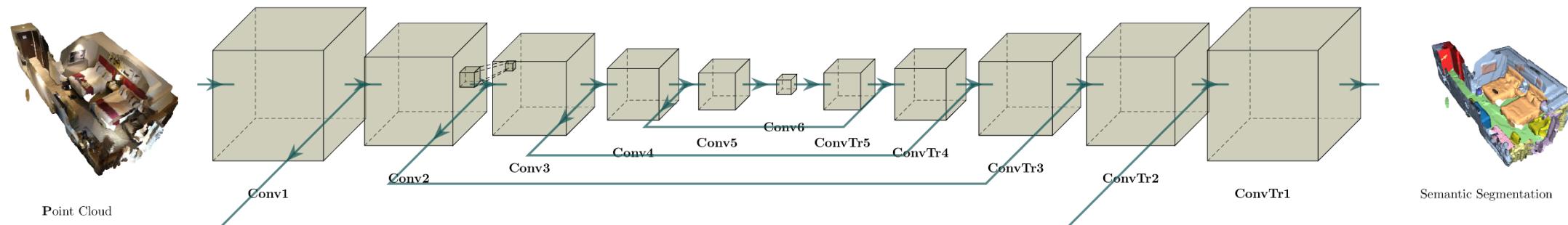
## » 2D vs. 3D

- 3D卷积直接应用到三维点云数据：将3D卷积核直接应用与三维点云数据，将时间轴t改为z轴特征提取，可以完美契合三维点云的相关应用。
- 缺陷：
  1. 最小的 $3 \times 3 \times 3$ 卷积，需要 $3^3$ 参数，etc:  $5^3$  ( $n^2 \rightarrow n^3$ )。
  2. 滑动窗口策略遍历全部空间场景。
  3. 网络参数规模增长。

如果需要处理的仅为单个室内场景/三维目标扫描，或许不需要考虑实时性问题，网络计算效率尚可接受。  
但是扩展到机器人、无人驾驶高度动态场景—**实时性约束显著！**

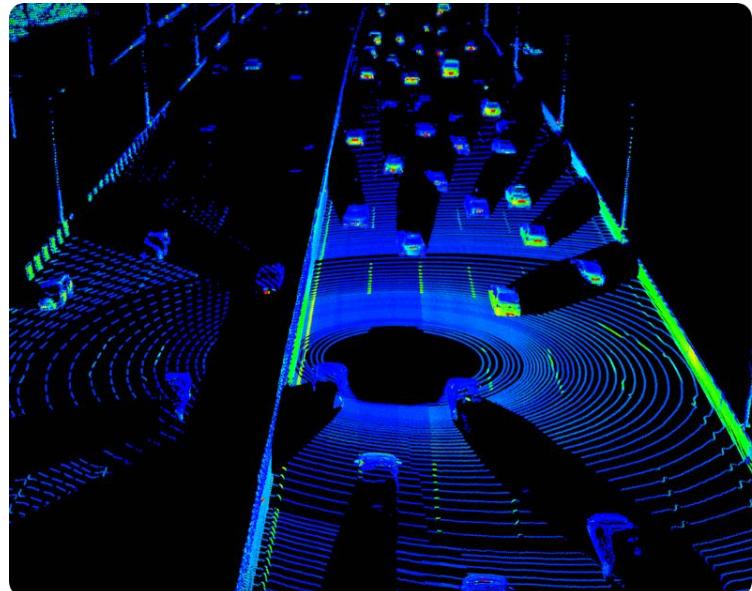


3D Unet架构  
的场景分割



# Part 5 | 卷积

## » 2D vs. 3D



### 缺陷

- 用常规3D卷积直接处理三维点云的低效性：
- 大量空白区域需要进行**无效计算**。
  - 空间点分布稀疏、**不规律**、**小卷积核采样低效**
  - 网络参数的学习和收敛所需的样本**更大**
  - 开展时序任务时，网络规模需要进一步提升。

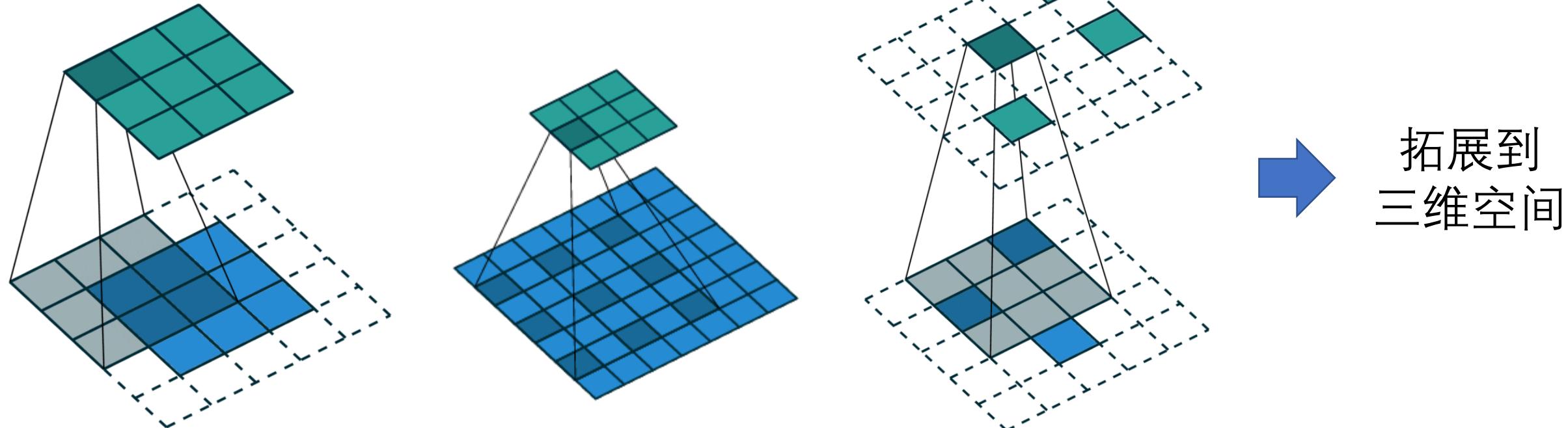
### 点云特性

Sensor	HDL-64E
	
Range	Up to 120m
Range Accuracy	Up to ±2 cm (Typical) <sup>4</sup>
# of Lines	64
Horizontal FoV	360°
Vertical FoV	26.9°
Horizontal Resolution	0.08° – 0.35°
Vertical Resolution	0.4°
Points Per Second (Single Return Mode)	~ 1,300,000
Points Per Second (Dual Return mode)	~ 2,200,000 <sup>5</sup>
Refresh Rate	5-20 Hz
Operating Voltage	12V - 32V
Power Consumption	60 W (Typical) <sup>2</sup>
Weight (without cabling)	~ 28 lbs. (12.7 Kg)
Operating Temp	-10°C to +60°C <sup>3</sup>
Storage Temp	-40°C to +85°C
Output	UDP packets over Ethernet
Ethernet Connection	100 Mbps
GPS Timesync	\$GPRMC
Laser	903nm Class 1 eye safe

### » 2D vs. 3D

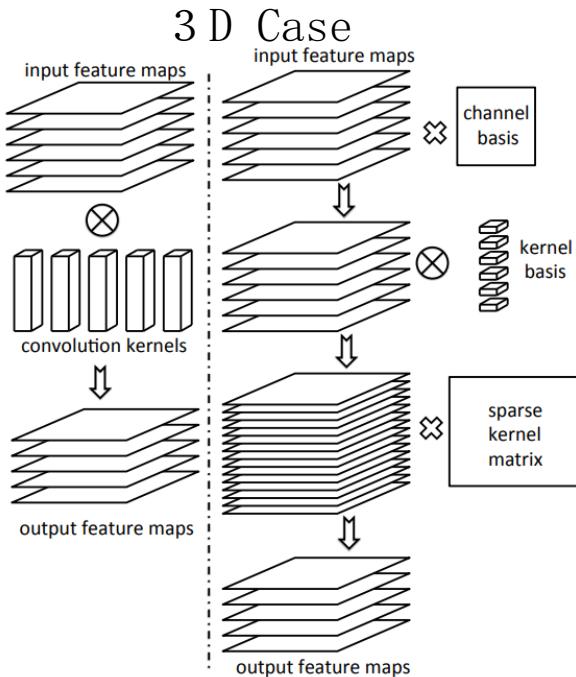
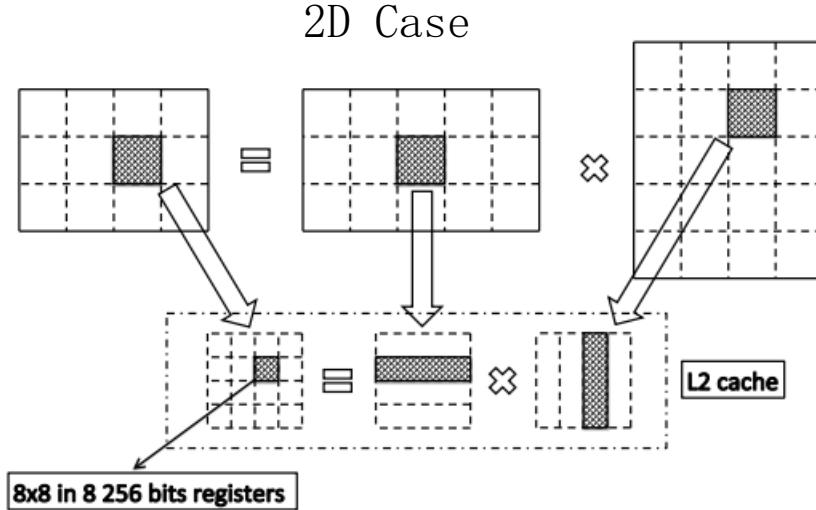
① 模型特征压缩，实质上等价于处理稀疏空间特征。

- 在2D卷积发展过程中，出现过drop out策略，即随机的丢弃一些网络层的部分特征（通过设置如dropout rate=10%），来提升网络的随机泛化能力和鲁棒性。
- 空洞卷积（相对普通卷积），则时直接设置固定的stride，用相同规模的卷积参数去提取更大的特征区域（感受野）。
- 稀疏卷积则是在编码时仅考虑具有特征的点。

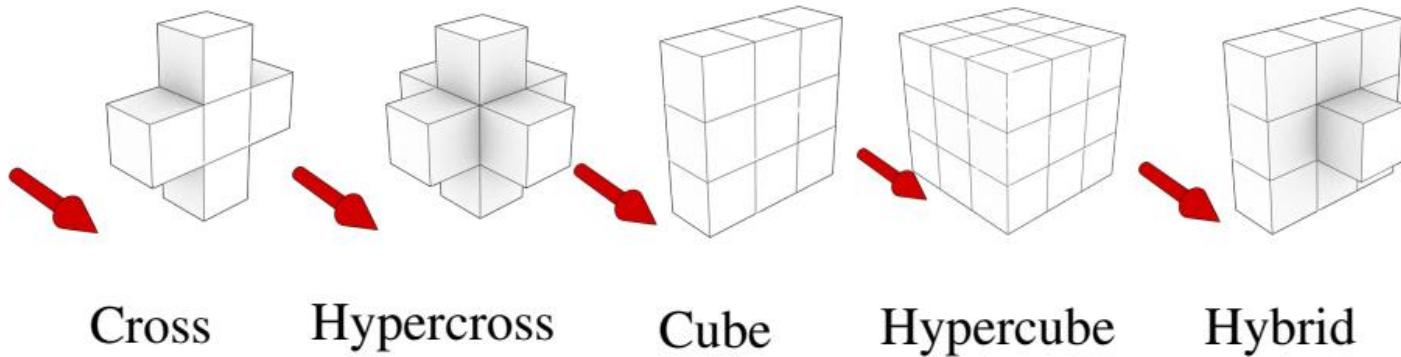


# Part 5 | 卷积

## » 2D vs. 3D



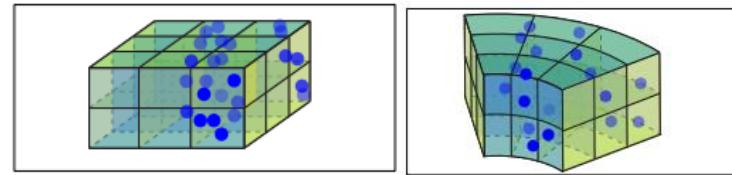
- ③ 异性变种核（高维）：考虑标准卷积核尺度内的不同特征抽取组合，可以设计小于标准卷积核的任意稀疏卷积核，实现稀疏卷积特征降维的目标。



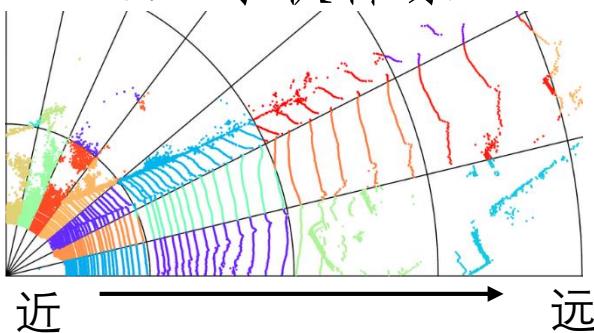
- ② 规则卷积拆分：MobileNet
  - $3 \times 3$  卷积拆分为  $1 \times 3$  和  $3 \times 1$  卷积的组合，通过层扩展还能增加网络深度，并降低  $1/3$  特征。
  - pointwise+channelwise 编码的组合
  - 同样适用扩展到三维空间。

# Part 5 | 卷积

## » 2D vs. 3D



- 体素化 (Voxelization) 能够降维特征规模、降低特征的随机性，结合稀疏编码可**发挥两者优势**。
- 常见Voxel-based策略（以传感器为中心）：
  1. Cube体素
  2. 柱状体素 ✓
  3. 球状体素



柱状体素：符合激光雷达空间采样规律：近稠密，远稀疏  
更加均匀的编码使得对原始点统计特性计算更合理。

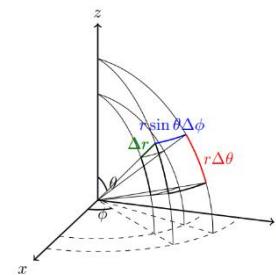
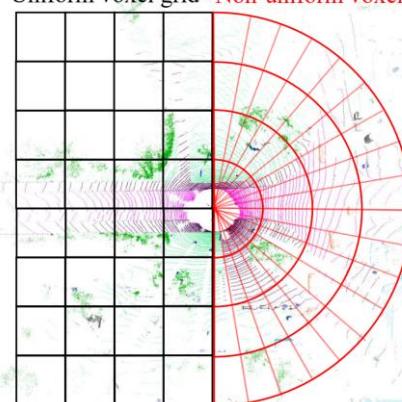


Fig. 3. Voxel in the spherical coordinate system. Its size is dependent on  $r$  and  $\theta$ .

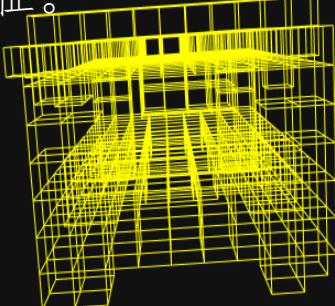
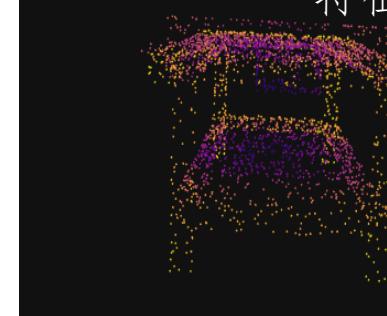
Uniform voxel grid Non-uniform voxel grid



实际上人的感知可以处理**高度抽象的粗粒度感知**信息。

VOXEL GENERATOR

按固定尺寸进行体素划分，用体素内点的统计特征代替原始点特征。



# Part 5 | 卷积

## » 逐点编码网络

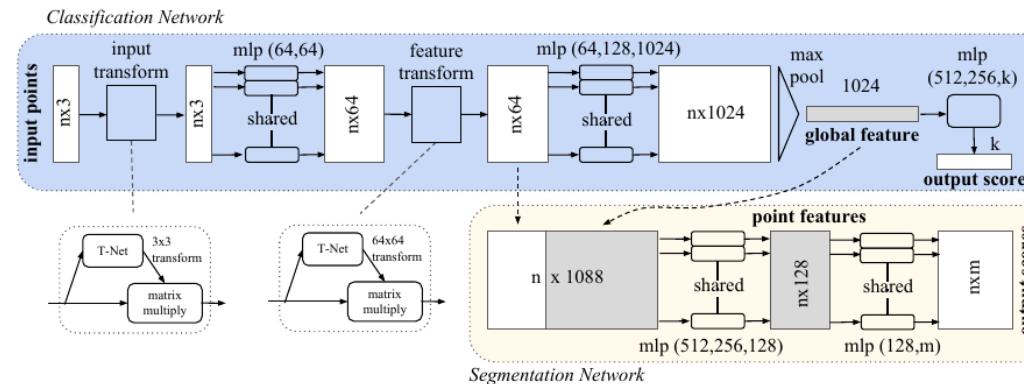
输入

$N \times 4$

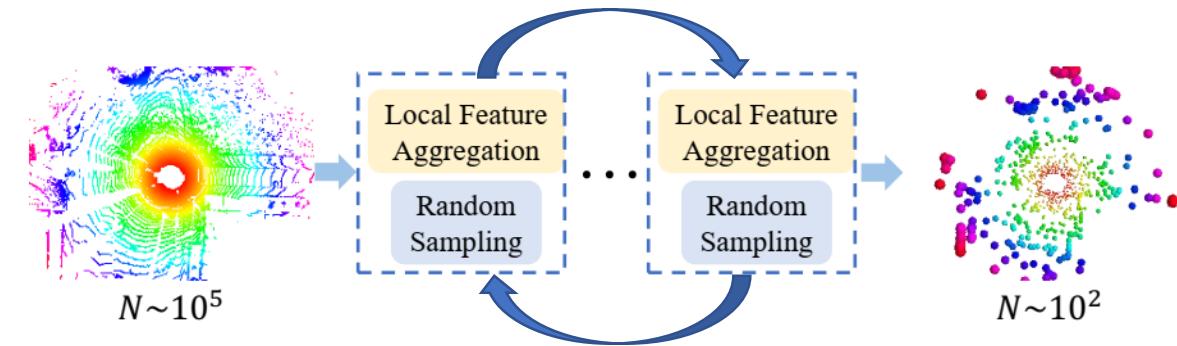
(x,y,z, i...)

...

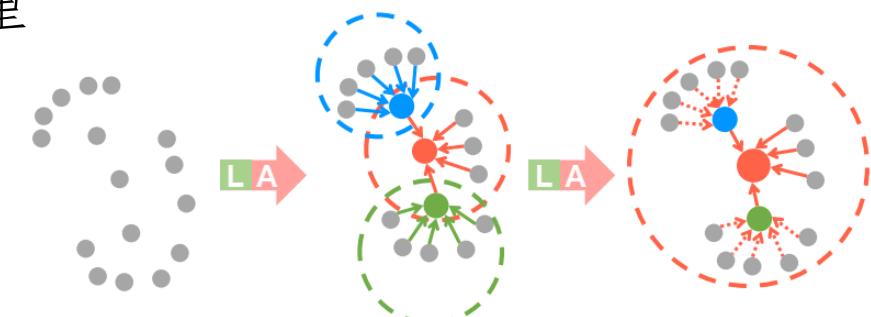
(x,y,z, i...)



- 既然点是无序的，直接对点进行逐点编码。
- 共享权重的MLP多层感知机，对原始点进行高维特征提取。
- 采用max-pooling进行特征聚合，相当于每个点的新特征会被聚合特征替代。



缺点：局部结构信息捕捉能力较弱。  
像上面的RandLA-Net就通过KNN搜索，  
来提升每个点周围特征的聚合；结合注  
意力机制来优化来自不同近邻点的特征  
权重

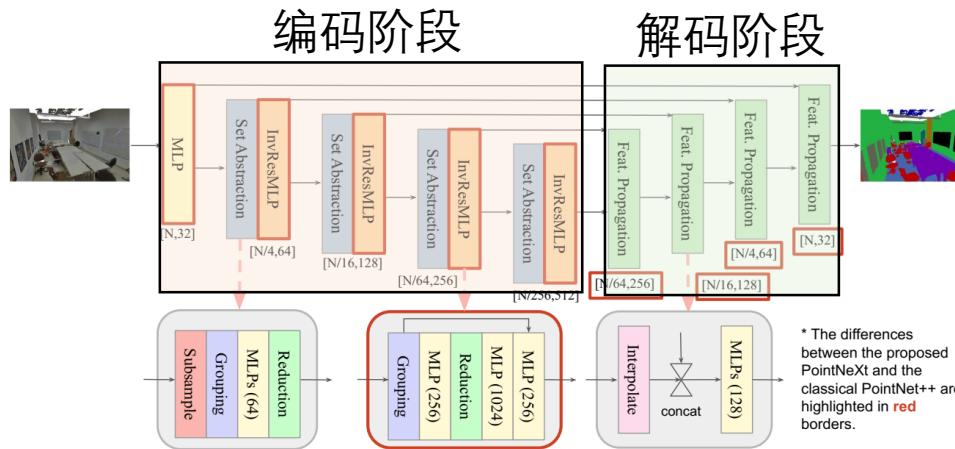


# Part 5 | 卷积

## » 点云感知任务建模

- 前面的内容，从点云数据结构、基础计算—3D卷积、数据采样模式等方面简要介绍了深度学习在点云处理中的一些基础。
- 以**语义分割**（Semantic Segmentation）任务为例，网络需要处理单帧点云，并预测逐点的语义标签，通常采用编解码（Encode-decode）结构的U-Net：将输入数据编码成紧凑的特征表示，再通过解码器将其还原或转化为目标输出。

PointNeXt



- 编码特征能够压缩数据、剔除冗余信息、并构建全局特征关联。
- 网络结构简单，特征图先逐渐缩小，再逐渐放大；并通过跨层联接构建浅层+深层信息关联。
- 可适应不同任务，且编解码特征尺度（scale）可控调节。

# Part 5 | 卷积

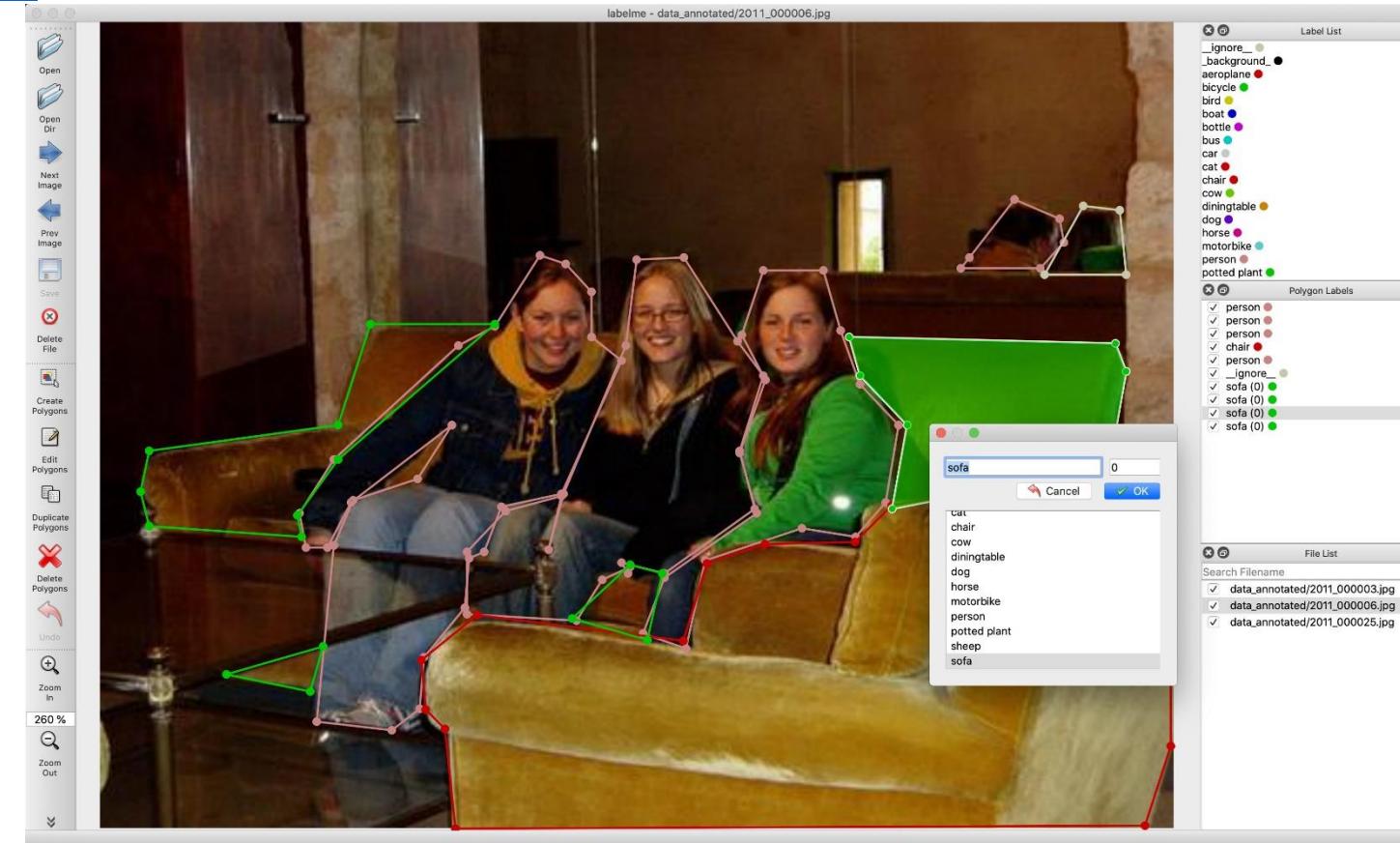
## » 数据标注

- 图像感知任务的数据标注: [labelme](#)
- 在2D平面进行标注
  - 矩形标定框
  - 分割掩码
  - trackID
  - 火柴人骨架

简单、直观、所见即所得、高效



延伸到三维点云任务？？？

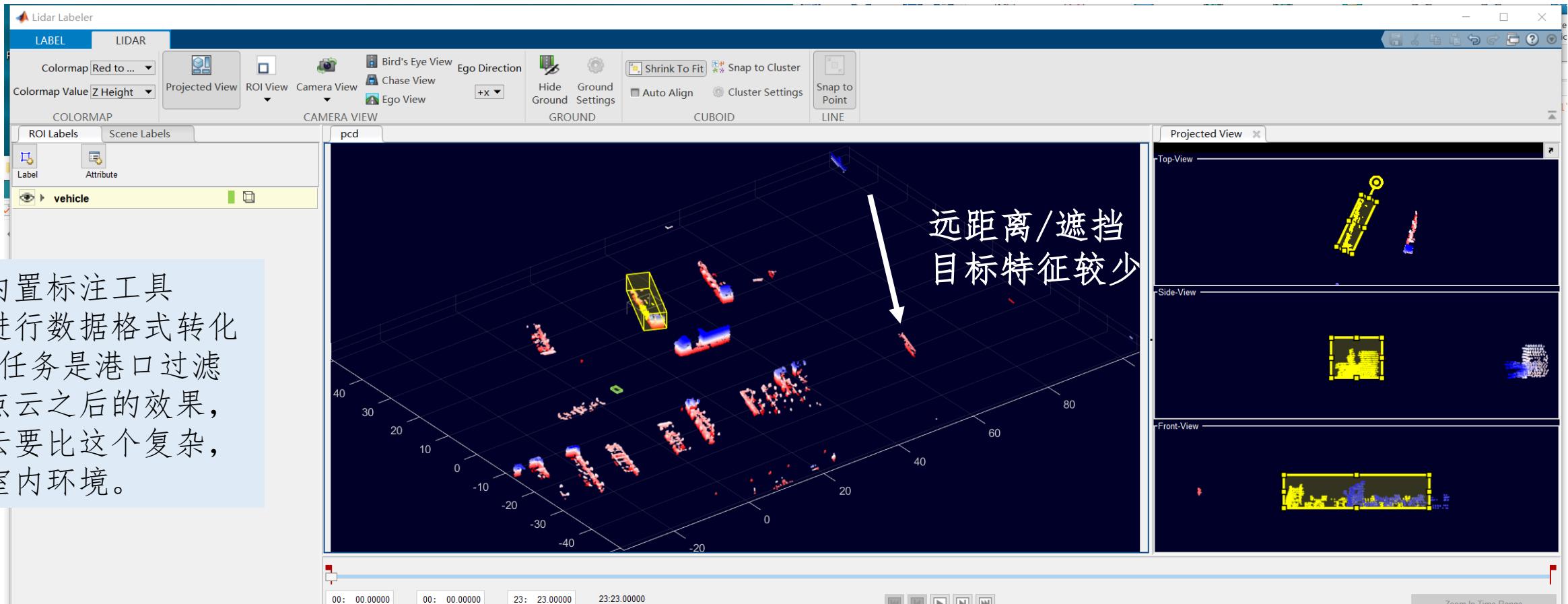


基于有监督训练的深度学习数据驱动还依赖合理的任务定义和精确的标签。

# Part 5 | 卷积

## » 数据标注

- 三维点云数据由于存在三维到二维显示界面的投影，存在视角转换问题，对于检测bbox和分割等任务通常需要借助：**鸟瞰图**、**侧视图**和**前视图**辅助投影观测，甚至相同视角图像辅助，以及人工多次调节，数据标注流程异常繁琐、效率低下。



Matlab内置标注工具  
但需要进行数据格式转化  
PS: 这个任务是港口过滤掉地面点云之后的效果，实际点云要比这个复杂，特别是室内环境。

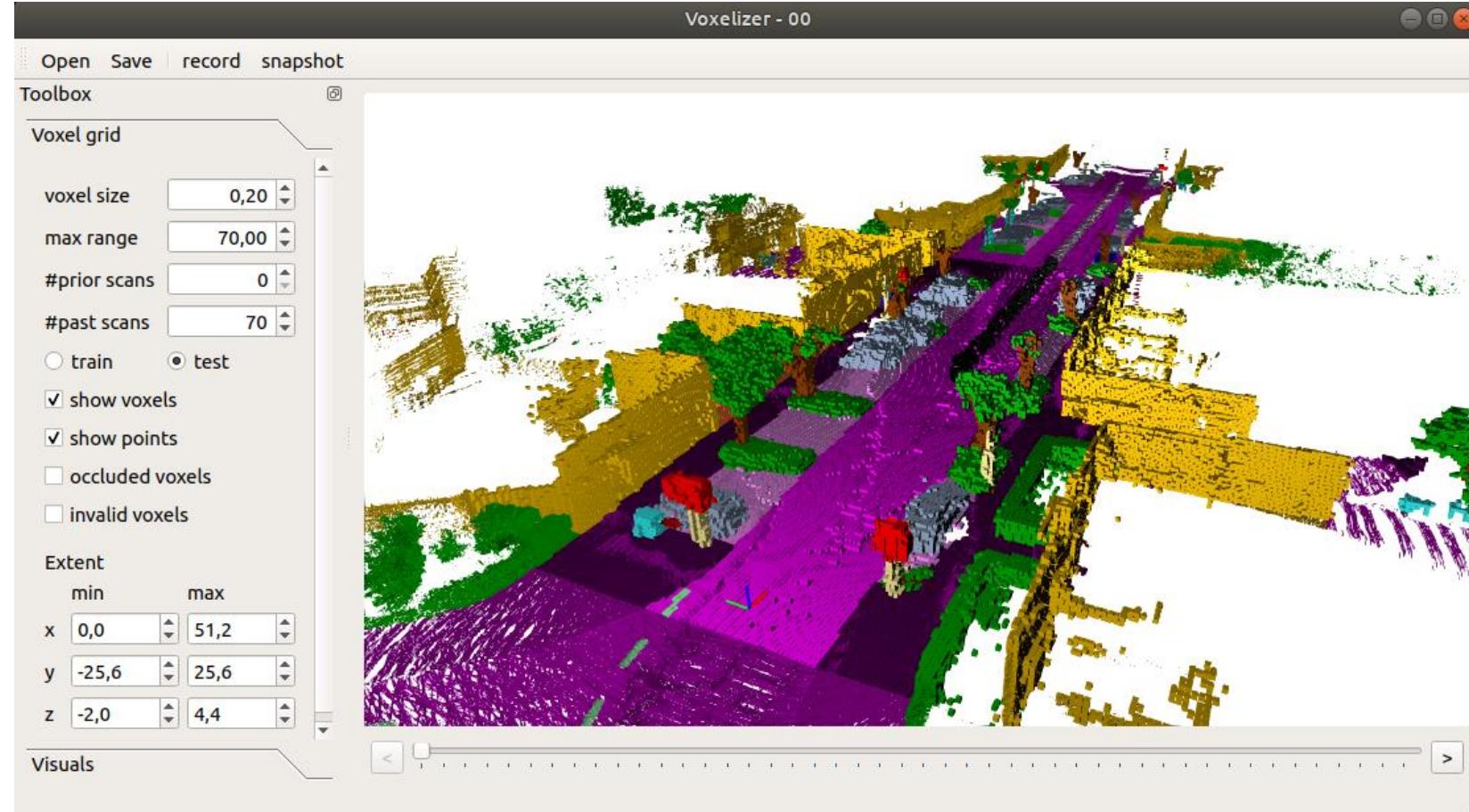
# Part 5 | PointLabeler

点云语义分割

## » 数据标注

通过**SLAM**将连续帧叠加，形成稠密点云地图，结合**人在环的位姿优化**，基于叠加的点云地图来弥补**单帧点云**中远距离目标、遮挡目标信息缺失的问题，能够有效增加远距离目标的特征。

- 提升目标特征—多帧叠加**互补缺失信息**。
- 提升标注效率--叠加标注能够实现**批处理**，**一次选择—多帧叠加标注**。



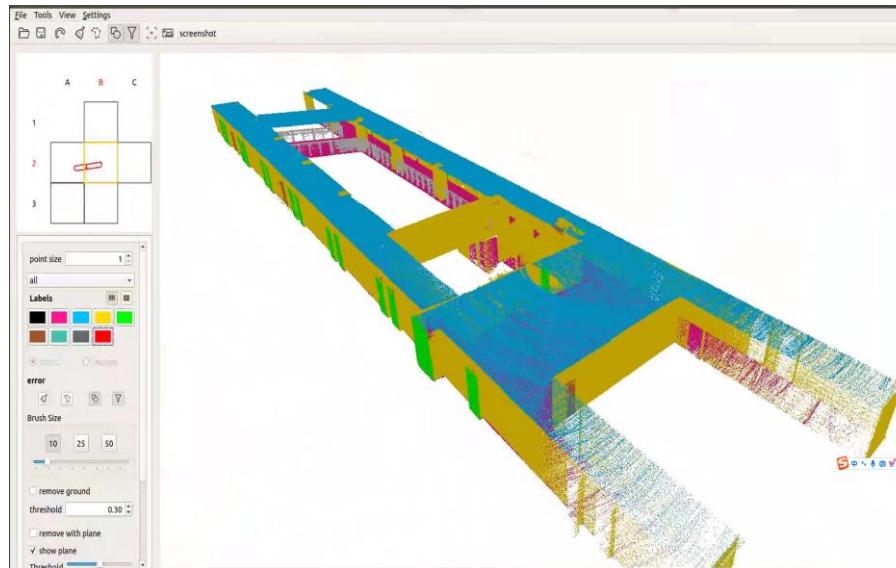
# Part 5 | PointLabeler

实验操作

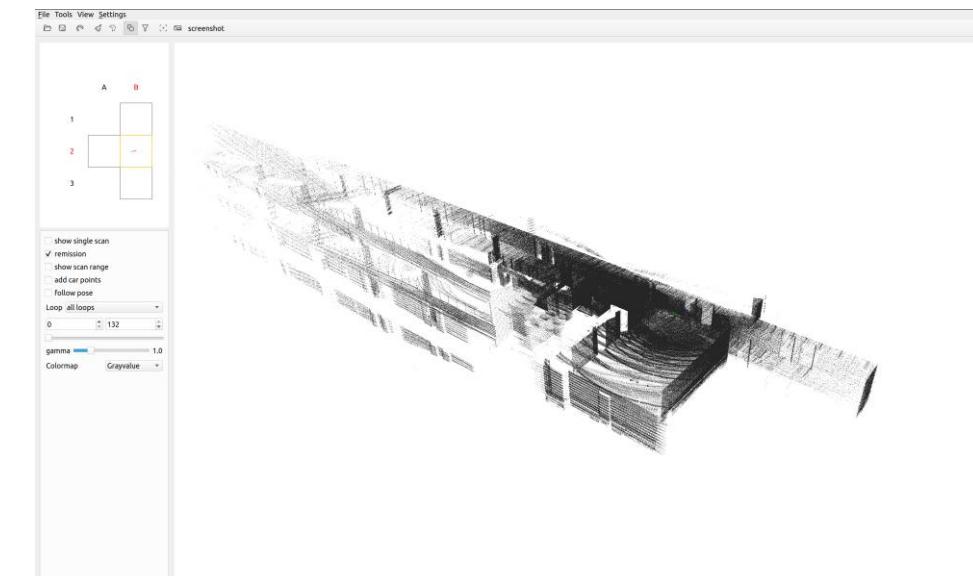
## » 数据标注

因为课程时间有限，并且使得每组同学都能够体验数据标注过程，我们将整个数据集进行了顺序拆分，分为10个subset。

2人一组使用PointLabeler完成数据标注工作，NAS中数据共有10组，按照签名顺序分组进行小型数据集标注。



完整的标注结果



其中一个子集叠加后的未标注点云

# Part 5 | PointLabeler

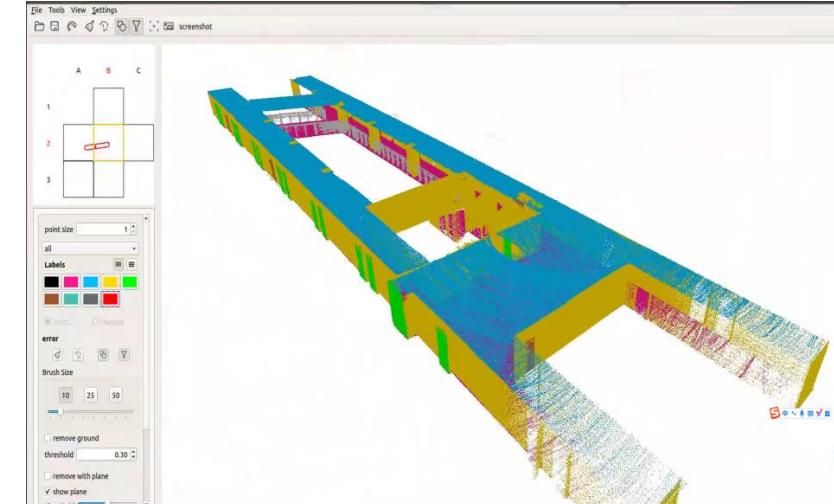
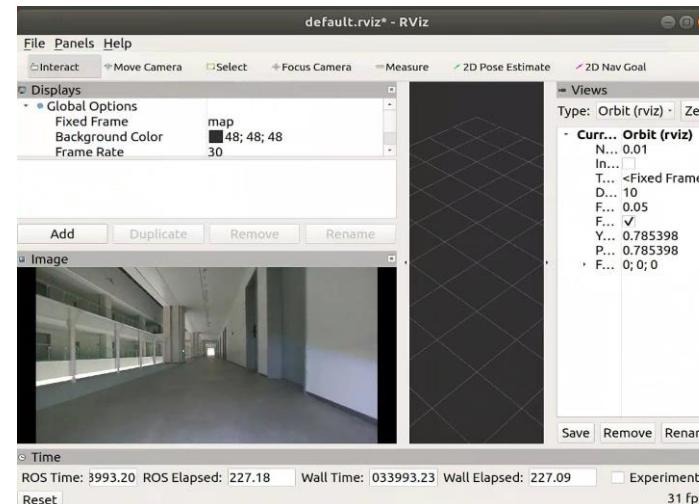
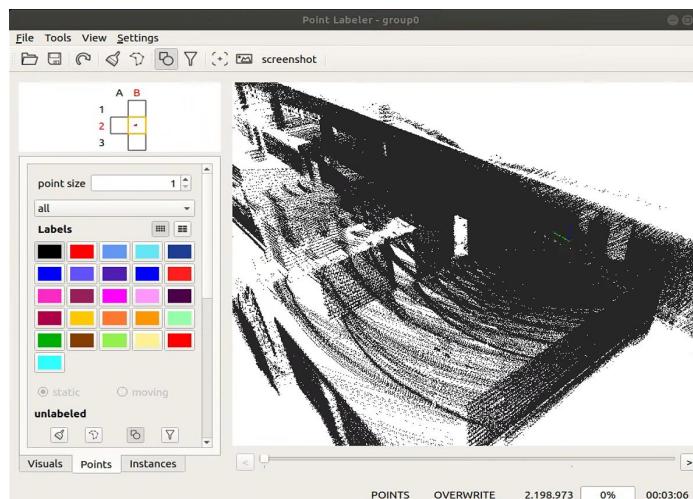
实验流程

## » 实验流程

配置软件环境  
下载未标注数据集  
载入未标注数据

划分标注类别  
ROSbag回放相机  
确定标注位置

进行标注  
学习软件使用方法  
及标注思路



# Part 5 | PointLabeler

环境安装

## » 开发环境配置-PointLabler

Github链接: <https://github.com/hahakid/3D-point-cloud-processing-and-visualization-practice.git>

### 依赖说明:

- Eigen >= 3.2
- boost >= 1.54
- QT >= 5.2
- OpenGL Core Profile >= 4.0

### 构建:

#### 1. 安装必要依赖项

```
$ sudo apt install git libeigen3-dev libboost-all-dev  
$ sudo apt install qtbase5-dev libglew-dev  
$ sudo apt install python-pip  
$ sudo pip install catkin_tools catkin_tools_fetch  
empty
```

#### 2. build

将点云课程github文件中的lecture4/point\_labeler.zip文件解压后:

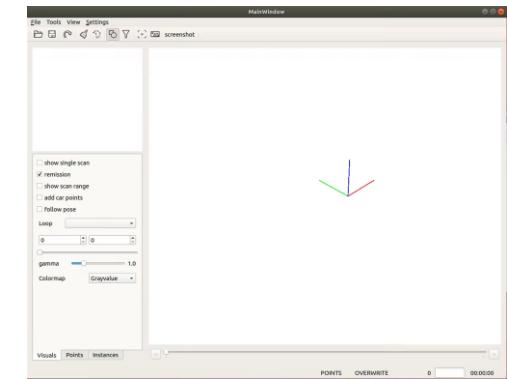
```
$ cd point_labeler  
$ cmake -S . -B build  
$ cmake --build build
```

#### 3. 运行测试

在point\_labeler/bin目录下在命令行中执行:

```
$ ./labeler
```

出现右图即安装成功



# Part 5 | PointLabeler

## 软件使用方法

### » 载入未标注数据

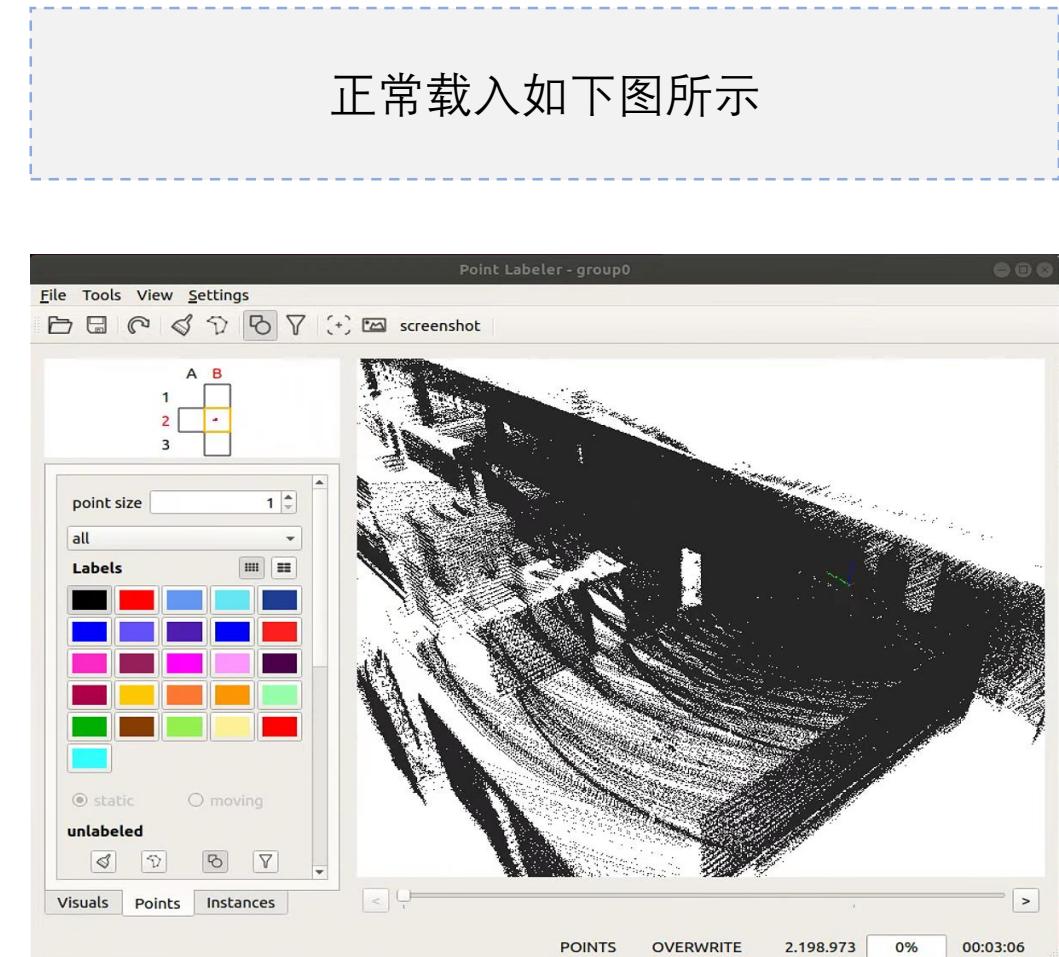
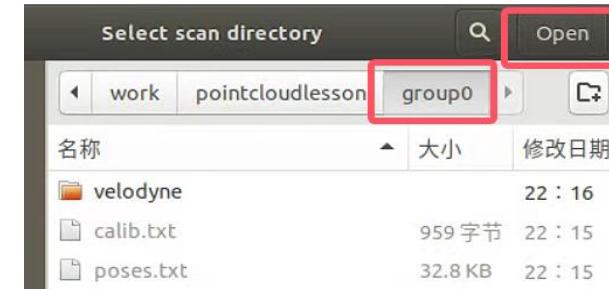
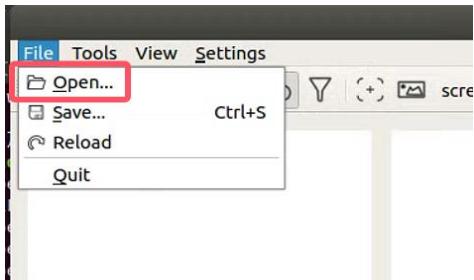
1. 大规模点云渲染会消耗显卡，建议在GPU性能较好的笔记本上进行标注。
2. 两个人只需要标注一次即可，无需重复标注（仅测试对比使用）。

#### 使用前配置：

1. 将NAS中的labels.xml文件下载下来，替换point\_labeler/bin目录下的labels.xml文件
2. 根据自己的电脑修改point\_labeler/bin目录下settings.cfg文件中gpu\_memory大小

#### 打开文件：

1. 下载NAS中data目录下对应group组别的未标注数据
2. 利用上页ppt的打开命令，打开point\_labeler图形化软件
3. 在point\_labeler中利用打开选项选中下载下来的未标注数据文件夹



# Part 5 | PointLabeler

软件使用方法

## » 划分标注内容

### 基于相机传感器确定场景语义类别

id	class	颜色	RGB	示例	备注
0	unlabeled		0 0 0	-	-
1	地面		255 20 147		-
2	天花板		0 191 255		天花板上的东西均无法辨别，均标注为ceiling
3	墙面		255 215 0		包括墙上的按钮，安全出口等无法辨别，统一归为墙面
4	门		0 255 0		包括玻璃门，电梯门和教室门
10	error point		255 0 0	从开着的门中反射到的户外，其他杂物等	较少且无法辨别的点

# Part 5 | PointLabeler

软件使用方法

## » ROSbag回放确定标注位置

1. 在命令行窗口中，输入roscore，启动ros

```
$ roscore
```

2. 再启动一个新的命令行窗口，用rosbag命今回放bag文件，raw32.bag可用绝对路径指定文件位置

```
$ rosbag play raw32.bag
```

A terminal window titled "kid@kid-seg: ~/Jia/dataset". It shows the command \$ roscore being run, followed by \$ rosbag play /home/kid/Jia/dataset/raw32.bag. A message [INFO] [1731033398.276920883]: Opening /home/kid/Jia/dataset/raw32.bag is displayed. Below this, it says "Waiting 0.2 seconds after advertising topics... done." and lists several log entries starting with "[RUNNING]".

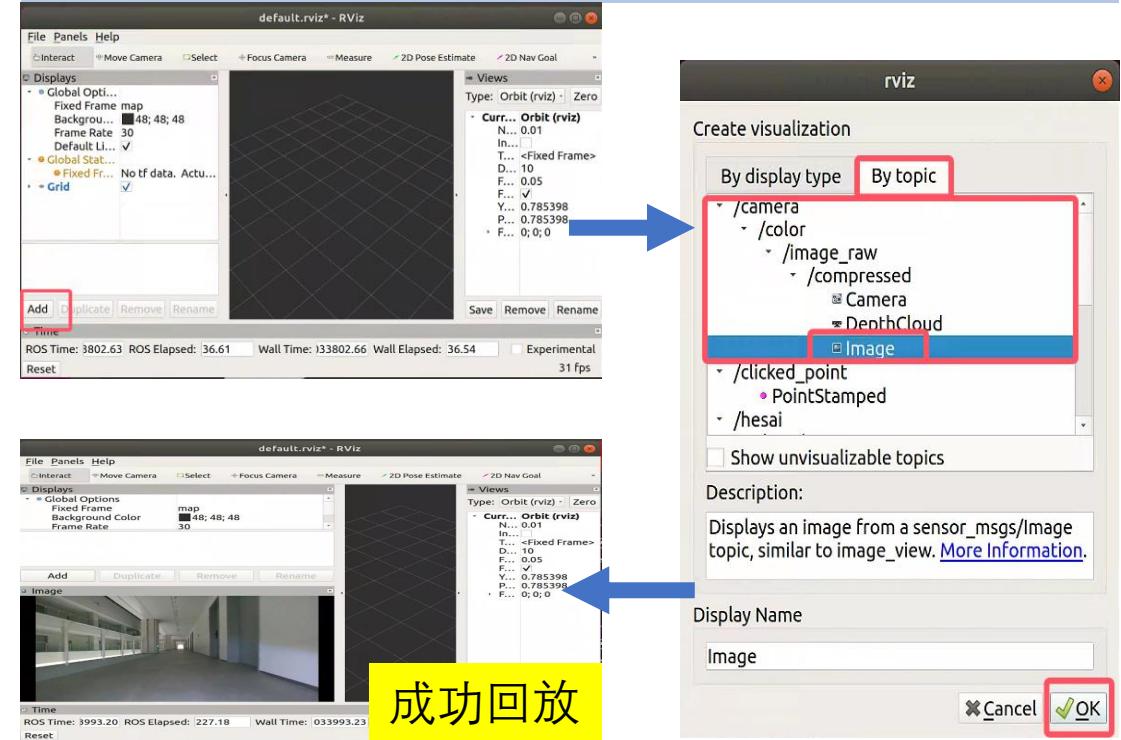
```
kid@kid-seg:~/Jia/dataset$ roscore
134.569289
kid@kid-seg:~/Jia/dataset$ rosbag play /home/kid/Jia/dataset/raw32.bag
[ INFO] [1731033398.276920883]: Opening /home/kid/Jia/dataset/raw32.bag
Waiting 0.2 seconds after advertising topics... done.

Hit space to toggle paused, or 's' to step.
[RUNNING] Bag Time: 1730221954.790334 Duration: 0.000000 /
[RUNNING] Bag Time: 1730221954.790408 Duration: 0.000073 /
[RUNNING] Bag Time: 1730221954.792318 Duration: 0.001984 /
[RUNNING] Bag Time: 1730221954.799728 Duration: 0.009394 /
[RUNNING] Bag Time: 1730221954.803277 Duration: 0.012942 /
[RUNNING] Bag Time: 1730221954.809764 Duration: 0.019430 /
[RUNNING] Bag Time: 1730221954.819719 Duration: 0.029385 /
[RUNNING] Bag Time: 1730221954.829871 Duration: 0.039537 /
[RUNNING] Bag Time: 1730221954.834059 Duration: 0.043725 /
[RUNNING] Bag Time: 1730221954.839731 Duration: 0.049396 /
[RUNNING] Bag Time: 1730221954.849686 Duration: 0.059352 /
[RUNNING] Bag Time: 1730221954.859696 Duration: 0.069362 /
```

3. 再启动一个新的命令行窗口，启动rviz

```
$ rviz
```

4. 选择add，通过topic添加相机的回放，topic选择

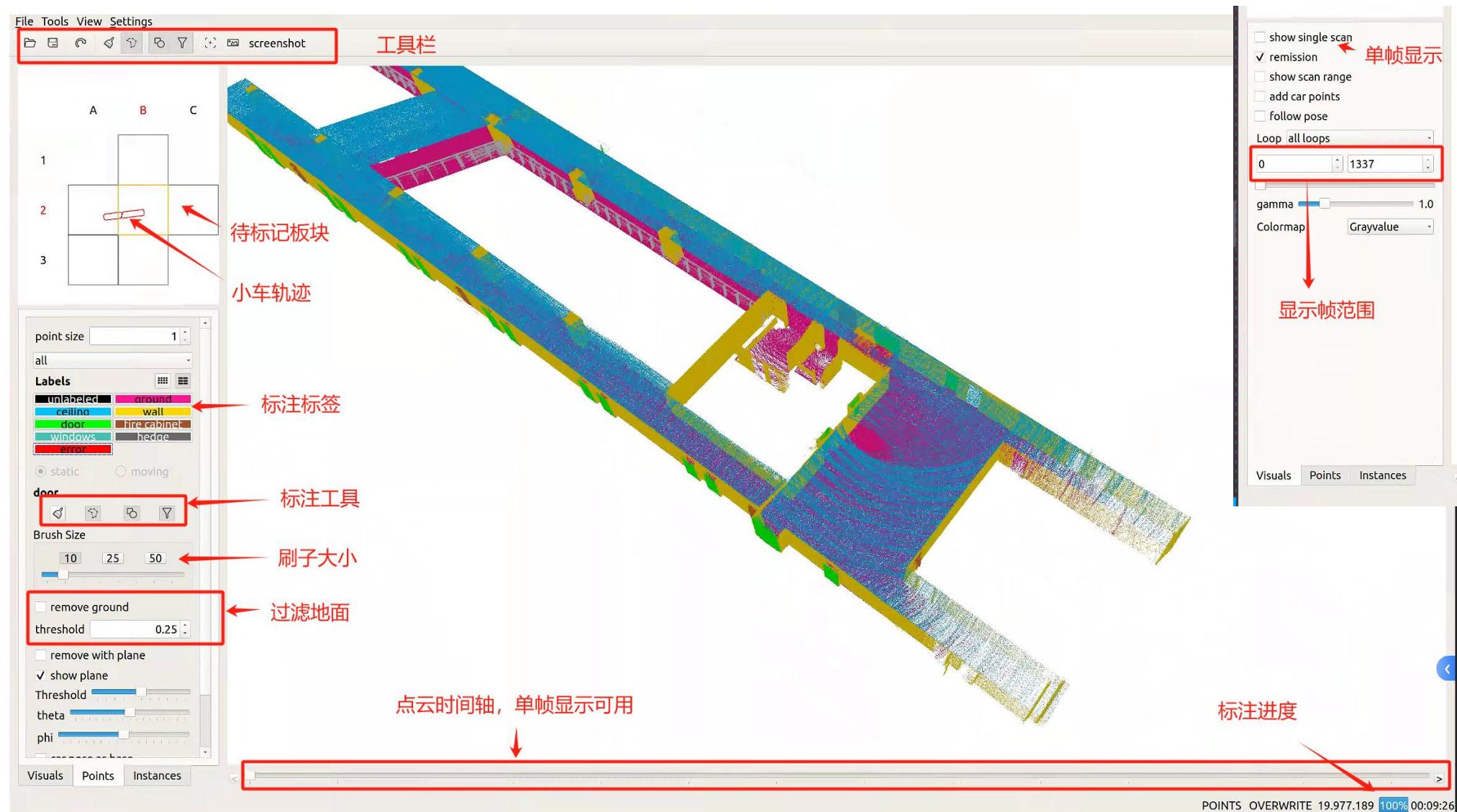


新终端快捷键：Ctrl+shift+T

# Part 5 | PointLabeler

软件使用方法

## » PointLabeler界面说明



### » PointLabeler配置文件自定义

可以修改point labeler/bin/目录下的label.xml文件，进行标签的种类自定义划分，增加或减少标签种类，序号颜色等

配置文件说明如右图所示

```
<config>
  <label>
    <id>0</id> → 标注序号
    <name>unlabeled</name>
    <description>all points with no special label.</description>
    <color>0 0 0</color>
    <root>unlabeled</root>
    <macro>unlabeled</macro>
    <category>unlabeled</category>
  </label>

  <label>
    <id>1</id> → 标准类别
    <name>ground</name>
    <description>Point cloud representing the ground.</description>
    <color>255 20 147</color>
    <root>ground</root> → 标注颜色
    <macro>ground</macro>
    <category>ground</category>
  </label>

  <label>
    <id>2</id>
    <name>ceiling</name>
    <description>Point cloud representing the ceiling.</description>
  </label>
```

标注分类，默认一种标签可分同一类

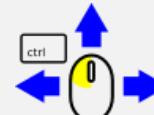
# Part 5 | PointLabeler

软件使用方法

## » 标注软件基本操作方法

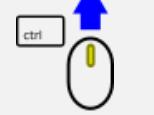
### 1. 移动视角

WASD、ctrl+鼠标左键



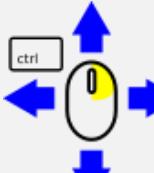
移动相机

ctrl+鼠标中键  
相机z轴



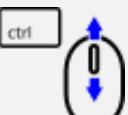
移动

ctrl+鼠标右键  
相机视角



移动

鼠标中间滚动  
缩放



相机

### 2. 工具栏



重新载入，未保存时慎用



刷子工具，用来给点云标注(着色)



多边形工具，用来给点云标注(着色)，鼠标左键构建多边形，右键确定，进行标注



覆盖模式，开启时标注，会覆盖已经标注的点云



筛选模式，开启时，按下ctrl键，选择点云标签，可过滤掉对应标签的点

ctrl+ 鼠标单击选中



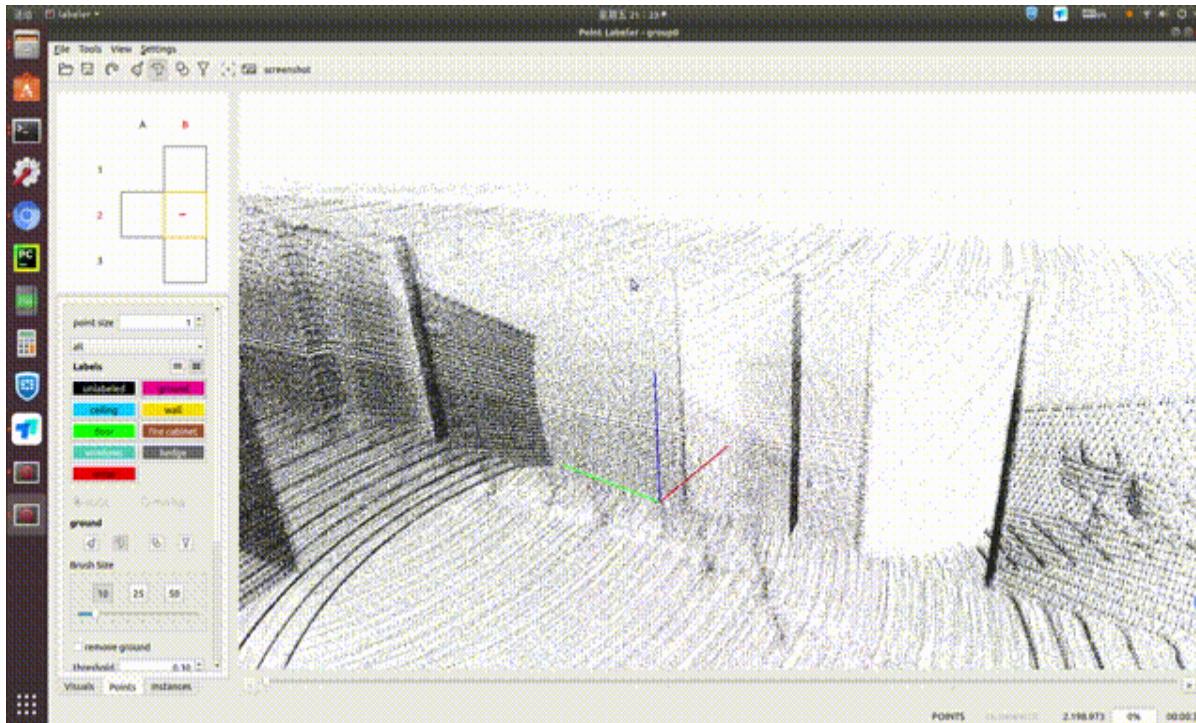
其他详细操作参阅nas中document目录下point\_labeler.wiki资料

# Part 5 | PointLabeler

软件使用方法

## » 标注软件基本使用

刷子工具 使用视频



多边形工具 使用视频



# Part 5 | PointLabeler

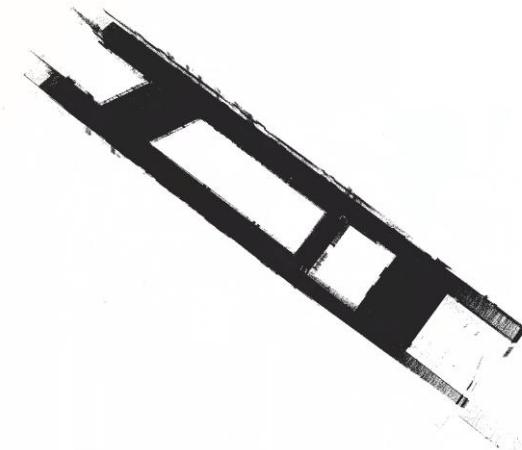
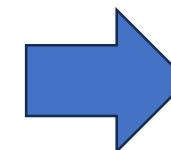
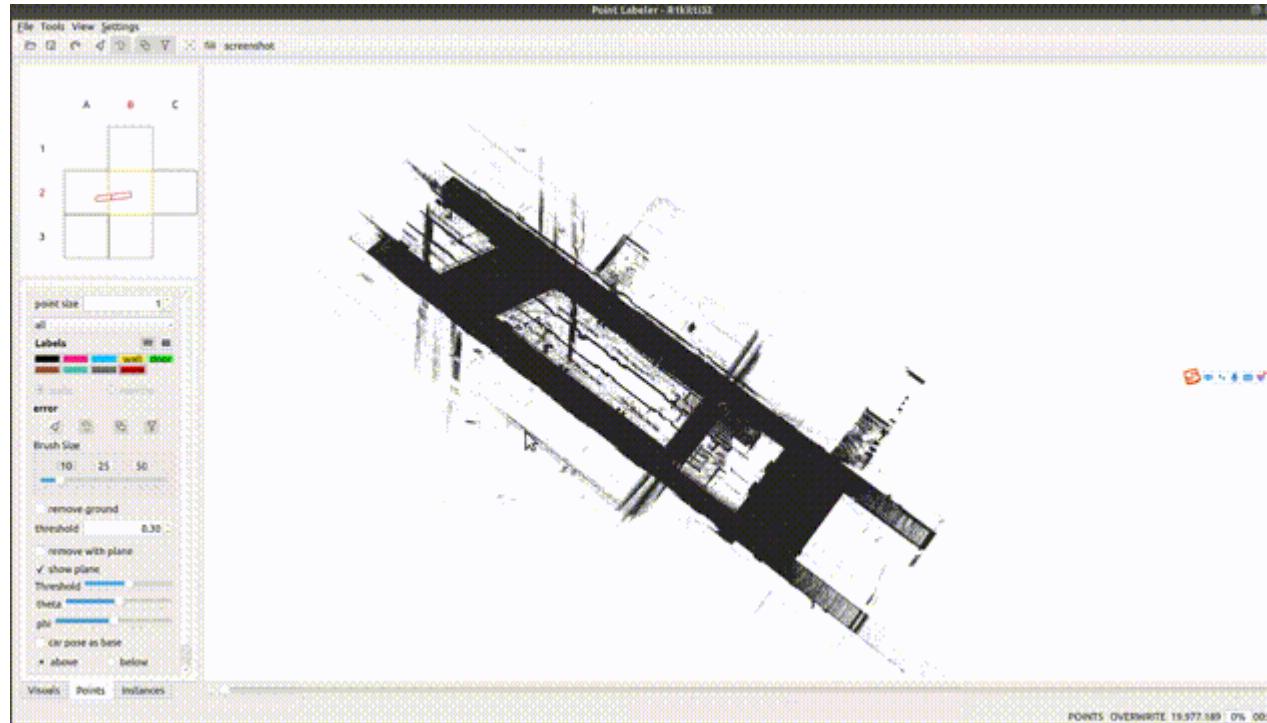
软件使用方法

## » 标注基本思路

- 1.标注采用先外后内的总体思路,先标注没有遮挡的物体.
- 2.然后将已标注的物体过滤(漏斗形状),标注没有遮挡的物体.
- 3.一层一层标注,直到标注完成.

- 1.先高度过滤地面以上的点, 然后标注地面
- 2.同样的方式处理天花板
- 3.过滤天花板和地面, 其它物体将孤立, 适合用多边形。

1 标记滤除外侧噪点



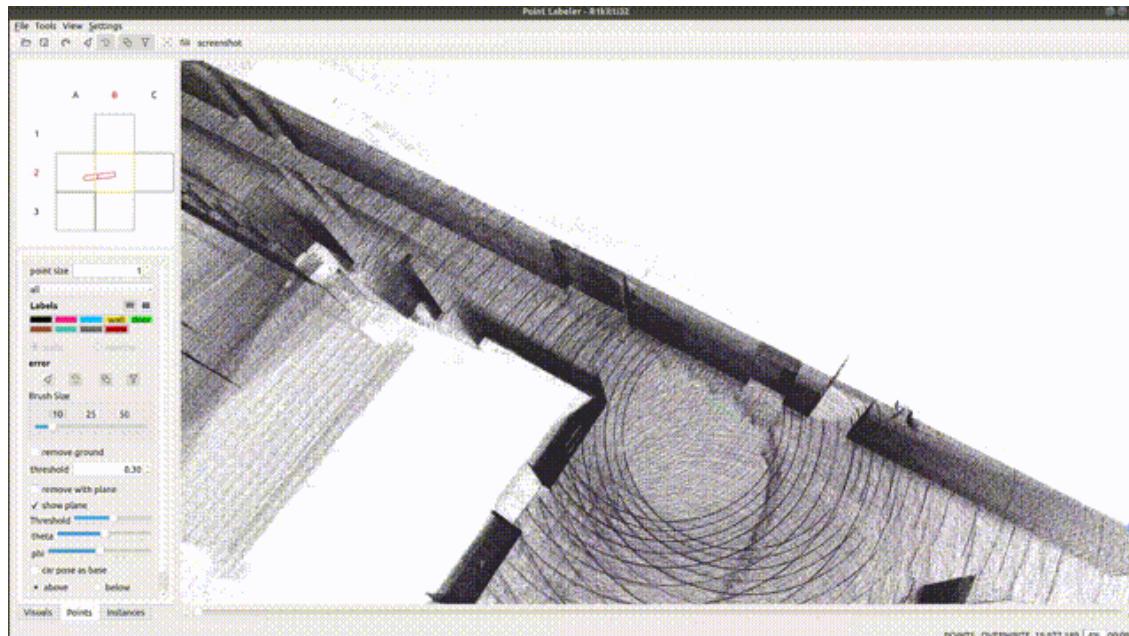
# Part 5 | PointLabeler

软件使用方法

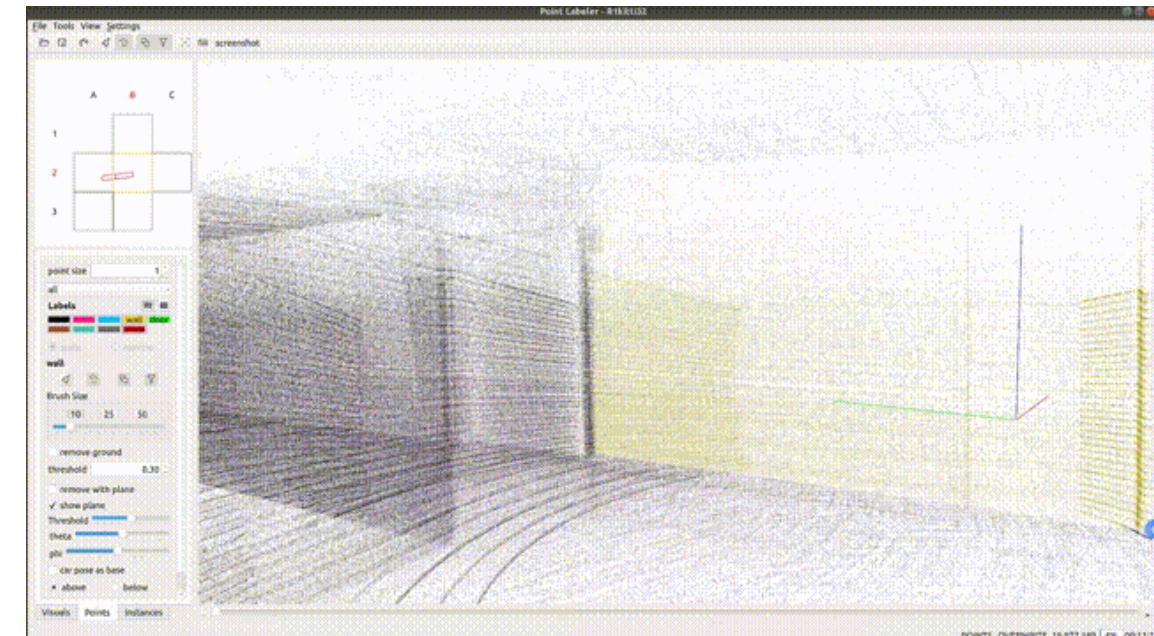
## » 标注基本思路

2: 标记外侧容易标记的点

示例：标记墙体视频



示例：标记门视频

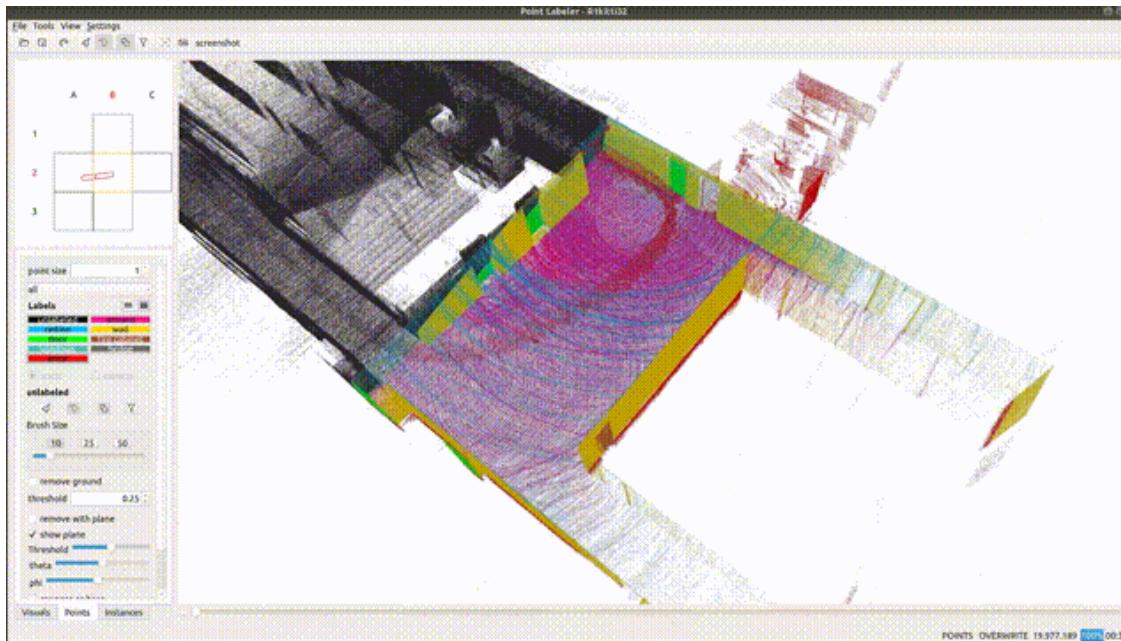


# Part 5 | PointLabeler

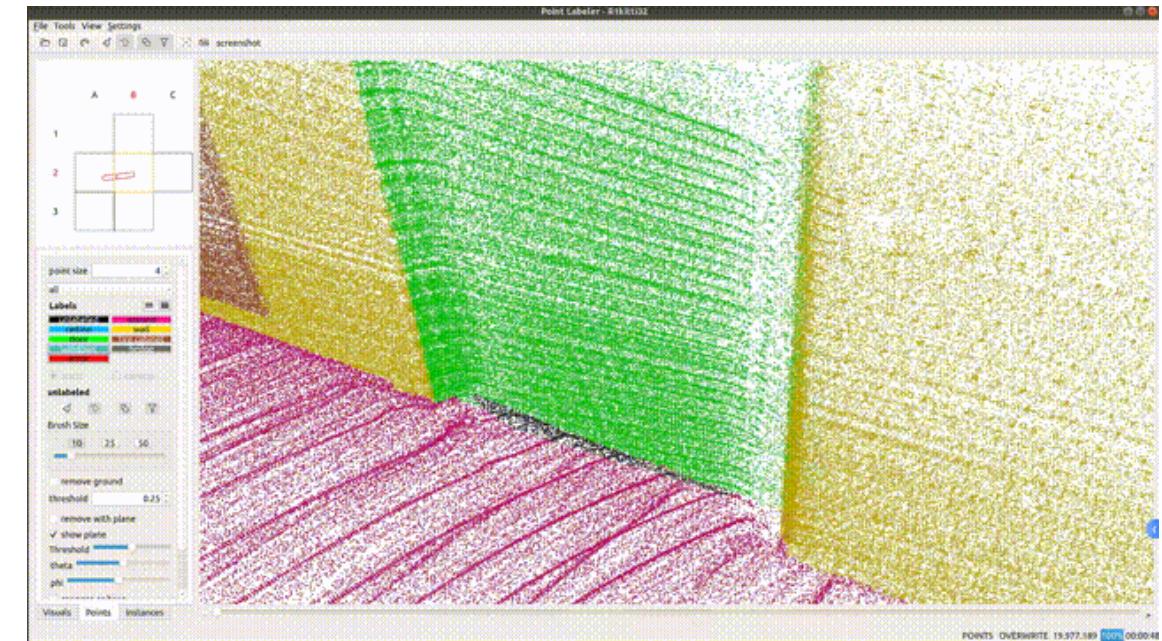
软件使用方法

## » 标注基本思路

示例：滤过已标记点



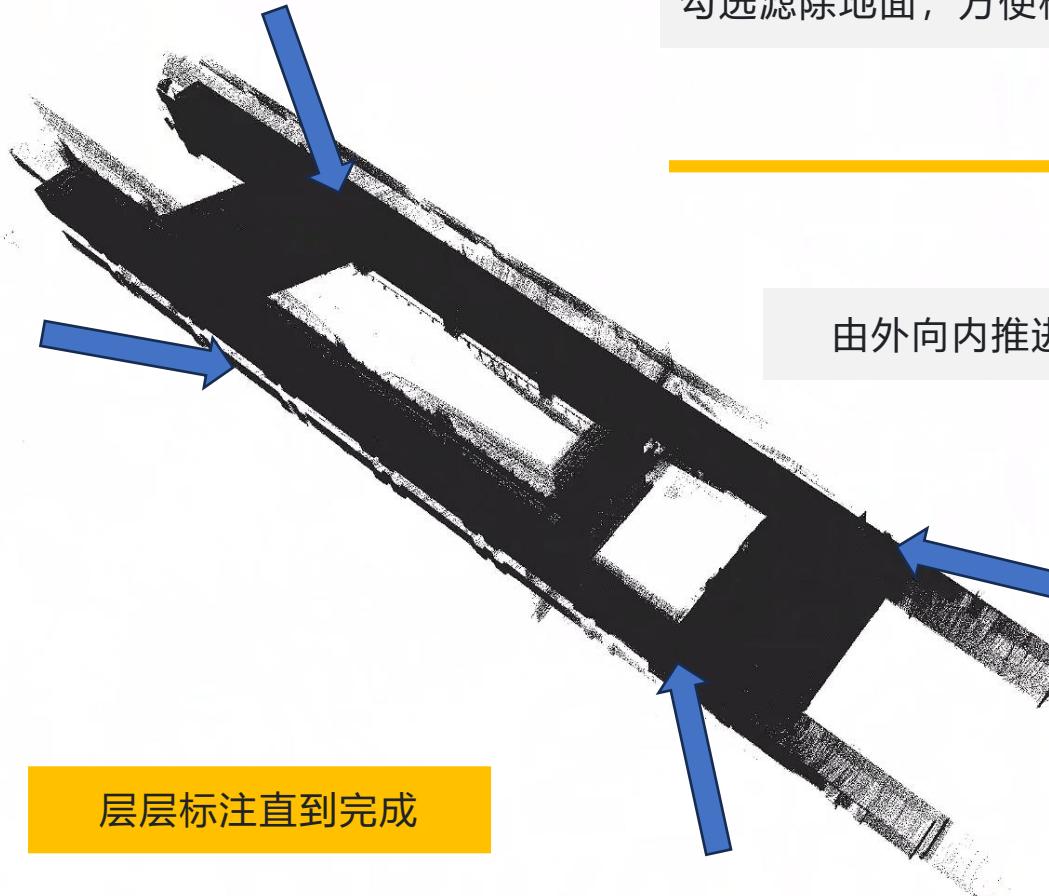
示例：利用覆盖模式优化细节



# Part 5 | PointLabeler

软件使用方法

## » 标注软件基本思路



勾选滤除地面，方便标记其他目标

由外向内推进

层层标注直到完成

remove ground

threshold

0.25

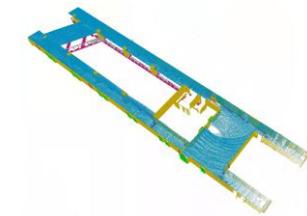


+ ctrl点击  
过滤标记点

unlabeled	ground
ceiling	wall
door	fire cabinet
windows	hedge
error	

标记一部分，过滤一部分

标注完成，细节优化



# Part 5 | Semantic segmentation

实验流程

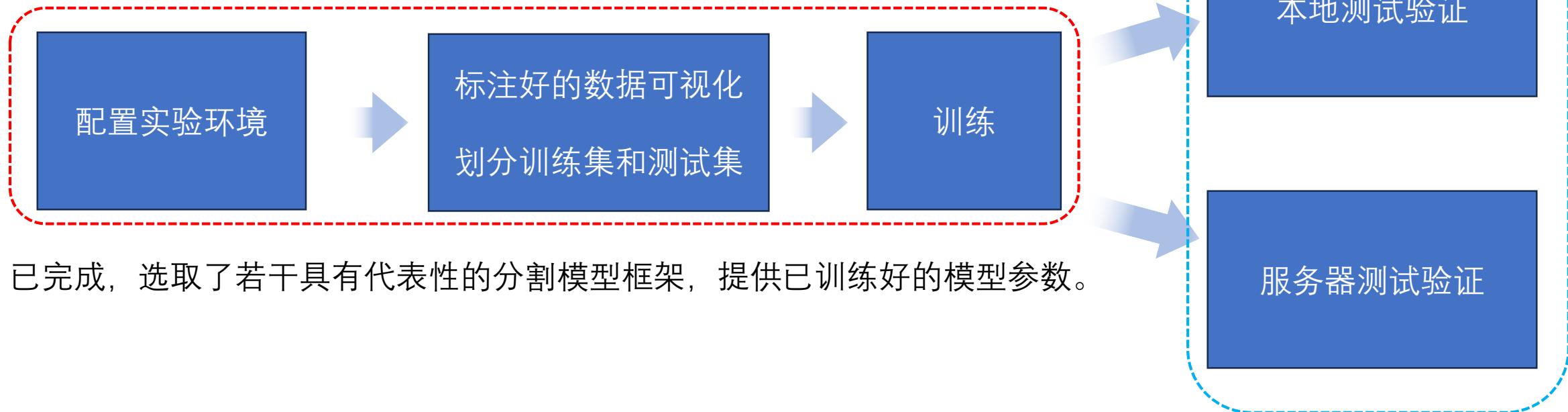
## » 实验目标:

对标注好的的科研1号楼数据进行点云语义分割训练，并进行测试评估及可视化验证。

这部分实验依赖GPU：

1. 建议有条件的可以在本地测试；
2. 远程实验室服务器测试，获取结果本地分析。

## » 点云语义分割实验流程



# Part 5 | Semantic segmentation

环境配置

## » 环境配置

### 1. 安装cuda和cudnn

Cuda下载链接 [CUDA Toolkit Archive | NVIDIA Developer](#)

1. 选择cuda11.3.1版本，选择对应的系统类型

CUDA Toolkit 11.3.1 (May 2021), Versioned Online Documentation

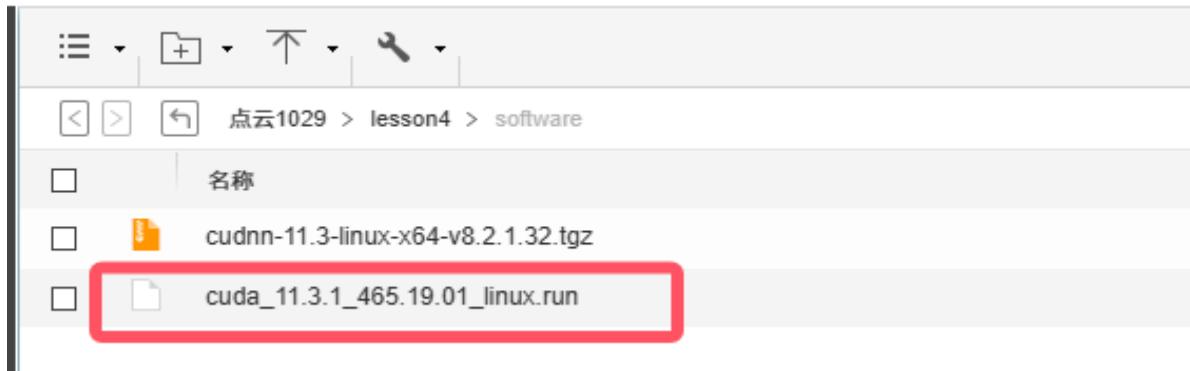
<b>Operating System</b>	Linux	Windows					
<b>Architecture</b>	x86_64	ppc64le	arm64-sbsa				
<b>Distribution</b>	CentOS	Debian	Fedor	OpenSUSE	RHEL	SLES	Ubuntu
<b>Version</b>	16.04	18.04	20.04				
<b>Installer Type</b>	deb (local)	deb (network)	runfile (local)				

# Part 5 | Semantic segmentation

环境配置

## » 环境配置

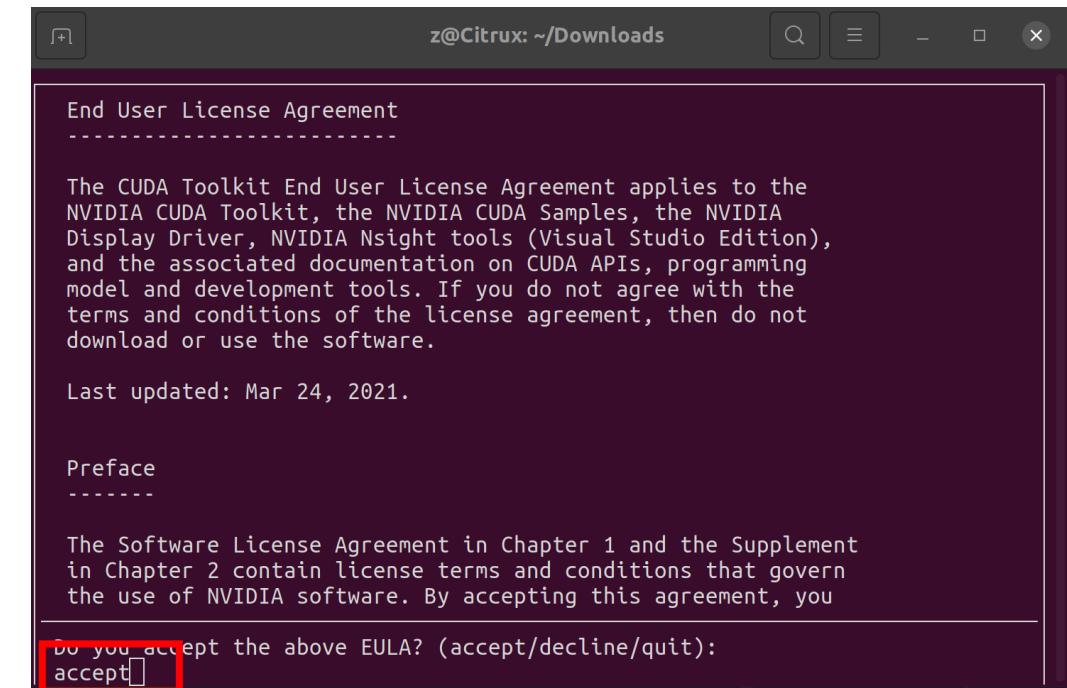
1. 如下载速度太慢，也可以在nas, software目录下，下载cuda



2. 进入本地下载目录，在命令行窗口中执行如下命令(sh 后为下载下来的文件名)

```
$ sudo sh cuda_11.3.1_465.19.01_linux.run
```

注：该cuda版本是x86\_64位系统架构，如是其他系统架构，则需要官网下载

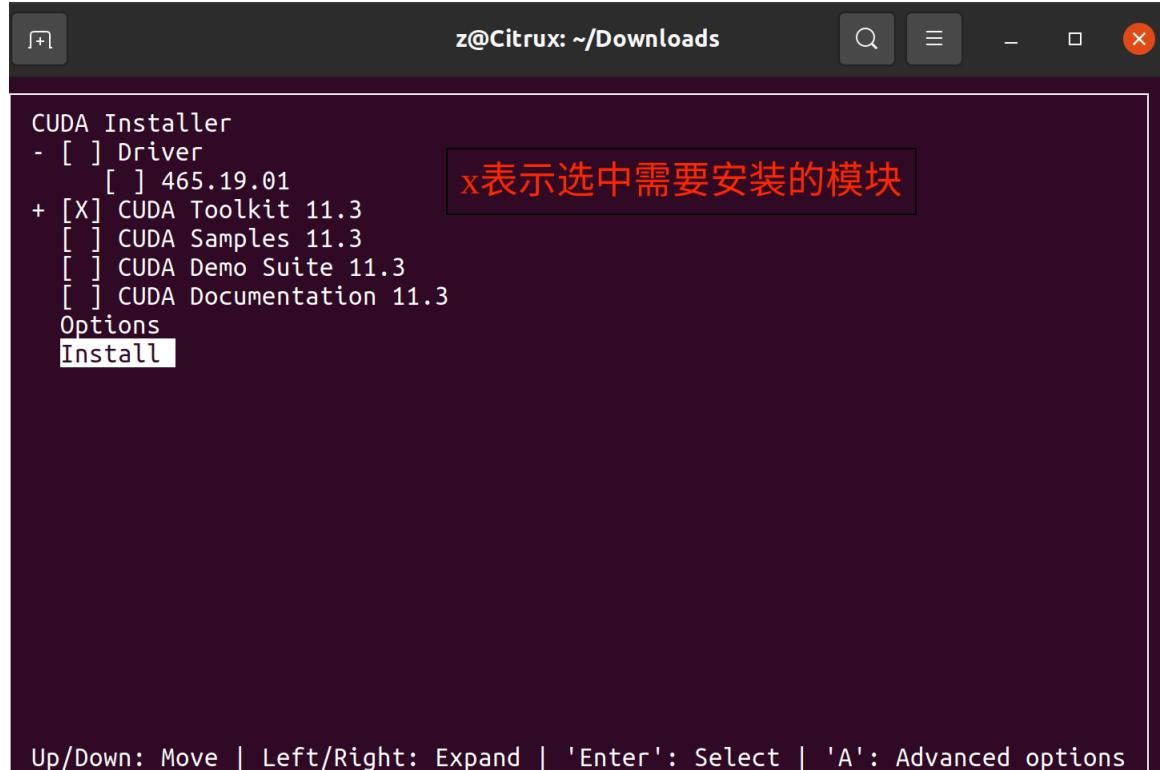


# Part 5 | Semantic segmentation

环境配置

## » 环境配置

2. 取消其他项目，只安装cudatoolkit



3. 等待安装，成功显示如下图信息

```
=====
= Summary =
=====

Driver: Not Selected
Toolkit: Installed in /usr/local/cuda-11.3/
Samples: Not Selected

Please make sure that
- PATH includes /usr/local/cuda-11.3/bin
- LD_LIBRARY_PATH includes /usr/local/cuda-11.3/lib64, or, add /usr/local/cuda-11.3/lib64 to /etc/ld.so.conf and run ldconfig as root

To uninstall the CUDA Toolkit, run cuda-uninstaller in /usr/local/cuda-11.3/bin
***WARNING: Incomplete installation! This installation did not install the CUDA Driver. A driver of version at least 465.00 is required for CUDA 11.3 functionality to work.
To install the driver using this installer, run the following command, replacing <cuda-installer> with the name of this run file:
sudo <cuda-installer>.run --silent --driver

Logfile is /var/log/cuda-installer.log
(base) z@Citrus:~/Downloads$
```

保证1.有显卡；2.使用了推荐的独显驱动；3.驱动版本高于cuda内置的。

# Part 5 | Semantic segmentation

环境配置

## » 环境配置

### 4. 配置环境变量

切换到home目录，使用vim或gedit打开.bashrc文件,复制下面的文本到.bashrc末尾

```
$ gedit .bashrc
```

```
# cuda
export CUDA_HOME=/usr/local/cuda-11.3
export
LD_LIBRARY_PATH=/usr/local/cuda11.3/lib64:LD_LIBRARY_PATH
export PATH=/usr/local/cuda-11.3/bin:$PATH
# cuda
```

```
139 if [ -f "/home/z/zyc/program/miniconda3/etc/profile.d/conda.sh" ]; then
140   . "/home/z/zyc/program/miniconda3/etc/profile.d/conda.sh"
141 else
142   export PATH="/home/z/zyc/program/miniconda3/bin:$PATH"
143 fi
144 fi
145 unset __conda_setup
146 # <<< conda initialize <<<
147 # =====
148
149
150 # cuda
151 export CUDA_HOME=/usr/local/cuda-11.3
152 export LD_LIBRARY_PATH=/usr/local/cuda11.3/lib64:LD_LIBRARY_PATH
153 export PATH=/usr/local/cuda-11.3/bin:$PATH
154 # cuda
155
156
157
158
159
```

执行：

```
$ source .bashrc
$ nvcc -V
```

显示如下信息即安装正常

```
(base) z@Citrux:~$ source .bashrc
(base) z@Citrux:~$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Mon_May_3_19:15:13_PDT_2021
Cuda compilation tools, release 11.3, V11.3.109
Build cuda_11.3.r11.3/compiler.29920130_0
(base) z@Citrux:~$ 
```

# Part 5 | Semantic segmentation

环境配置

## » 环境配置

### 2. 安装cudnn

cudnn下载链接

<https://developer.nvidia.com/rdp/cudnn-archive>

选择8.2.1版本

[Download cuDNN v8.2.4 \(September 2nd, 2021\), for CUDA 10.2](#)

[Download cuDNN v8.2.2 \(July 6th, 2021\), for CUDA 11.4](#)

[Download cuDNN v8.2.2 \(July 6th, 2021\), for CUDA 10.2](#)

[Download cuDNN v8.2.1 \(June 7th, 2021\), for CUDA 11.x](#)

[Download cuDNN v8.2.1 \(June 7th, 2021\), for CUDA 10.2](#)

[Download cuDNN v8.2.0 \(April 23rd, 2021\), for CUDA 11.x](#)

[Download cuDNN v8.2.0 \(April 23rd, 2021\), for CUDA 10.2](#)

[Download cuDNN v8.1.1 \(February 26th, 2021\), for CUDA 11.0,11.1 and 11.2](#)

[Download cuDNN v8.1.1 \(February 26th, 2021\), for CUDA 10.2](#)

[Download cuDNN v8.1.0 \(January 26th, 2021\), for CUDA 11.0,11.1 and 11.2](#)

选择cuDNN Library for Linux



[cuDNN Library for Linux \(aarch64sbsa\)](#)

[cuDNN Library for Linux \(x86\\_64\)](#)

[cuDNN Library for Linux \(PPC\)](#)

[cuDNN Library for Windows \(x86\)](#)

[cuDNN Runtime Library for Ubuntu20.04 x86\\_64 \(Deb\)](#)

[cuDNN Developer Library for Ubuntu20.04 x86\\_64 \(Deb\)](#)

[cuDNN Code Samples and User Guide for Ubuntu20.04 x86\\_64 \(Deb\)](#)

[cuDNN Runtime Library for Ubuntu20.04 aarch64sbsa \(Deb\)](#)

# Part 5 | Semantic segmentation

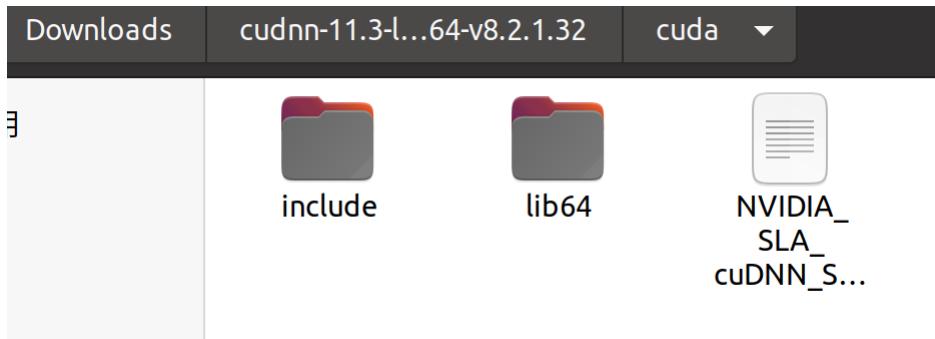
环境配置

## » 环境配置

1. 同cuda如下载速度太慢，也可以在nas, software目录下，  
下载cudnn

```
$ sudo cp lib64/libcudnn* /usr/local/cuda-11.3/lib64/  
$ sudo cp include/cudnn.h /usr/local/cuda-11.3/include/
```

2. 切换至下载目录，解压下载的cudnn文件，得到下列文件



3. 如图所示目录下打开终端，复制文件

4. 修改对应的文件权限

```
$ sudo chmod 777 /usr/local/cuda-11.3/lib64/libcudnn*
```

```
$ sudo chmod 777 /usr/local/cuda-11.3/include/cudnn.h
```

# Part 5 | Semantic segmentation

环境配置

## » 环境配置

### 3. 安装pcseg

#### 1. 创建conda环境

```
$ conda create -n pcseg python=3.8
```

#### 2. 激活conda环境

```
$ conda activate pcseg
```

#### 3. 安装pytorch

```
$ conda install pytorch==1.10.0 torchvision==0.11.0 cudatoolkit=11.3 -c pytorch -c conda-forge
```

#### 4. 安装必要的库

```
$ pip install nuscenes-devkit -i https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple
```

```
$ conda install pytorch-scatter -c pyg
```

```
$ sudo apt-get install libsparseshash-dev
```

# Part 5 | Semantic segmentation

环境配置

## » 环境配置

### 3. 安装pcseg

Github链接:

<https://github.com/hahakid/3D-point-cloud-processing-and-visualization-practice.git>

### 5. 将github下载的 /lesson4/pcseg32.zip文件解压

```
$ cd pcseg32/package  
$ mkdir torchsparse_dir/  
$ unzip sparsehash.zip  
$ unzip torchsparse.zip  
$ mv sparsehash-master/ sparsehash/
```

### 6. 设置主路径, 注意此处\${ROOT}要替换成OpenPCSeg所在路径 (整条复制, 先修改路径, 再复制)

```
$ cd sparsehash/  
.configure --prefix=/${ROOT}/pcseg32/package/torchsparse_dir/sphash/
```

# Part 5 | Semantic segmentation

环境配置

## » 环境配置

### 3. 安装pcseg

### 7. 构建torchsparse

```
$ make  
$ make install  
$ cd ..  
$ pip install ./torchsparse
```

构建需要一定的时间，成功后会显示如下信息：

```
Preparing metadata (setup.py) ... done  
Building wheels for collected packages: torchsparse  
  Building wheel for torchsparse (setup.py) ... done  
    Created wheel for torchsparse: filename=torchsparse-1.4.0-cp38-cp38-linux_x86_64.whl size=8862609 sha256=d1ce5b27a1766fd1c715ef14b06fff83c179a5f018d9de858cfb5912db0adcdd  
    Stored in directory: /tmp/pip-ephem-wheel-cache-bi3y9y5u/wheels/cc/1f/69/d959f8c93d695da2d6f2569cde9b789971bf8dff05e4e47f0  
Successfully built torchsparse  
Installing collected packages: torchsparse  
Successfully installed torchsparse-1.4.0
```

### 8. 构建Range Image Library

```
$ unzip range_lib.zip  
$ cd range_lib/  
$ python setup.py install
```

构建成功后会显示如下信息：

```
Installed /home/z/zyc/program/miniconda3/envs/pcseg/lib/python3.8/site-packages/rangelib-1.0.0-py3.8-linux-x86_64.egg  
Processing dependencies for rangelib==1.0.0  
Finished processing dependencies for rangelib==1.0.0
```

# Part 5 | Semantic segmentation

环境配置

## » 环境配置

### 3. 安装pcseg

### 8. 其他软件包

```
$ pip install pyyaml easydict numba torchpack strictyaml llvmlite easydict scikit-image tqdm SharedArray  
prettytable opencv-python open3d fastapi -i https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple  
  
$ pip uninstall setuptools  
$ pip install setuptools==59.5.0
```

### 9. 注册pcseg

```
$ python setup.py develop
```

# Part 5 | Semantic segmentation

实验流程

## » 划分测试集和训练集

下载NAS,或github下载的代码包，打开utils目录下的  
build\_train\_data.py 文件

```
target_path = "/media/z/Share/study/dataset/train_data"  
source_path = "/media/z/Share/study/dataset/R1kitti32"
```

1.修改target\_path目录，target目录为训练集和测试集划分好的输出目录

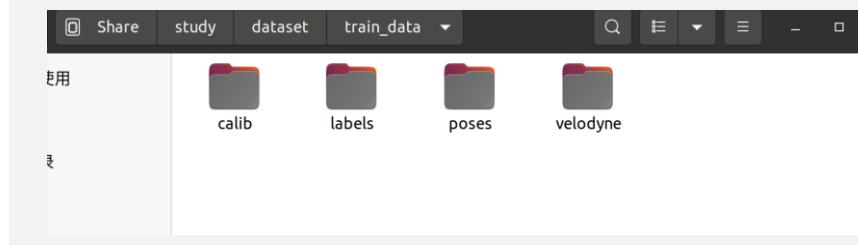
2.修改source\_path目录,source目录为从nas中下载的R1kitti32数据集的存放位置。

3.运行代码

代码运行成功输出如下信息

```
构建中,请等待...  
构建已完成, 输出路径: /media/z/Share/study/dataset/train_data  
进程已结束, 退出代码为 0
```

在对应目录生成velodyne labels 等数据集文件



# Part 5 | Semantic segmentation

实验流程

## » 标注好的数据可视化

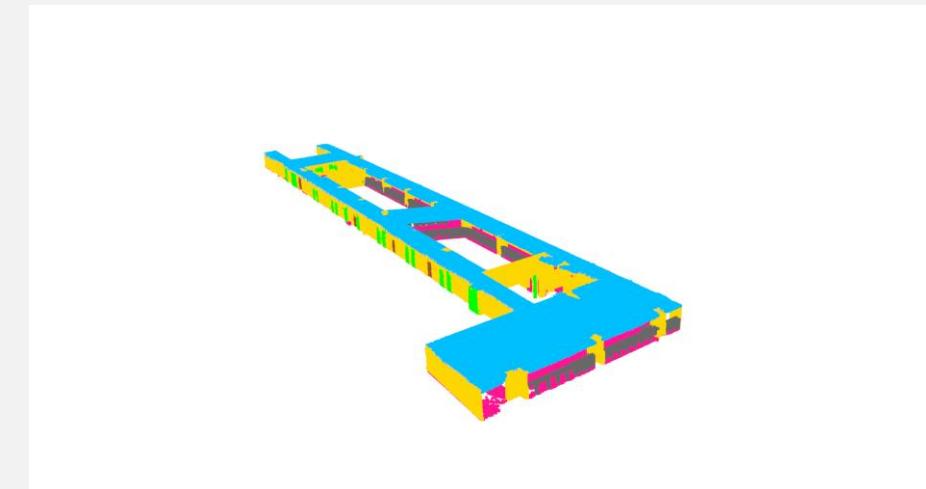
下载NAS,或github下载的代码包，打开utils目录下的add\_frame\_utils.py文件

1.修改同目录下的R132.yaml文件中root\_path目录，该目录为从NAS中下载下来的R1kitti32数据集或自己标注完成的group数据集所在位置。

2.根据需求修改down\_sample\_rate,下采样率，此变量为体素大小，调的越大，占用显存越小，推荐值：0, 0.2, 0.4, 0.6, 0.8

3.运行程序

运行成功即可看到标注的可视化效果



# Part 5 | Semantic segmentation

实验流程

## » 划分测试集和训练集

下载NAS,或github下载的代码包，打开utils目录下的  
build\_train\_data.py 文件

```
target_path = "/media/z/Share/study/dataset/train_data"  
source_path = "/media/z/Share/study/dataset/R1kitti32"
```

1.修改target\_path目录， target目录为训练集和测试集划分好的输出目录

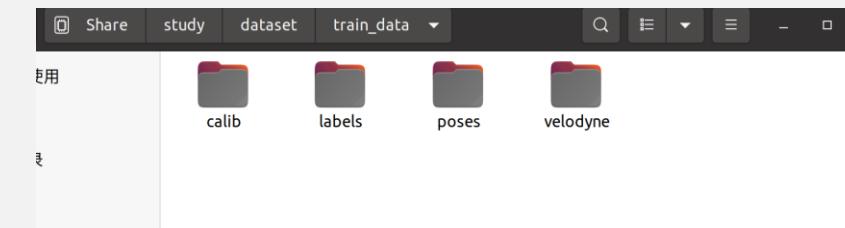
2.修改source\_path目录,source目录为从nas中下载的  
R1kitti32数据集的存放位置。

3.运行代码

代码运行成功输出如下信息

```
构建中,请等待...  
构建已完成,输出路径: /media/z/Share/study/dataset/train_data  
进程已结束,退出代码为 0
```

在对应目录生成velodyne labels 等数据集文件



# Part 5 | Semantic segmentation

实验流程

## » 本地训练 (可选)

计算机有独立显卡且显存比较大，可用使用本地训练：

1. 下载NAS中或github代码包，打开pcseg32文件夹。
2. 修改/tools/cfgs/voxel/semantic\_kitti/目录下 cylinder\_cy480\_cr5\_r1.yaml文件中

```
DATA_PATH: '/media/z/Share/study/dataset/train_data' # h
```

```
EVALUATION:  
    is_save_label: True  
    save_path: /media/z/Share/study/dataset/labelpredict
```

data\_path为上一步骤代码构建的数据集的位置  
save\_path为预测标签的保存路径

3. 在pcseg32根目录下打开终端，(先切换pcseg虚拟环境)  
执行：

```
$ python train.py --cfg_file  
tools/cfgs/voxel/semantic_kitti/cylinder_cy480_cr5_r1.  
yaml
```

成功启动训练如下图所示：

```
(logits): Conv3d(64, 20, kernel_size=(3, 3, 3))  
(change_dim): Sequential(  
    (0): Linear(in_features=64, out_features=256, bias=True)  
    (1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_st  
ats=True)  
    (2): LeakyReLU(negative_slope=0.01)  
)  
(point_logits): Linear(in_features=256, out_features=20, bias=True)  
(criterion_losses): Losses(  
    (ell_loss): ELLLoss()  
    (dice_loss_v0): DiceLossV0()  
    (dice_loss_v1): DiceLossV1()  
    (wce_loss): WeightedCrossEntropyLoss()  
    (cce_loss): CrossEntropyLoss()  
    (focalloss): FocalLoss()  
    (eqLv2): EQLv2()  
    (groupsoftmax): GroupSoftmax()  
    (groupsoftmax_fgbg): GroupSoftmax_fgbg_2()  
)  
(loss_funcs): CrossEntropyLoss()  
)  
2024-11-12 14:52:04,686 INFO Model parameters: 15.662 M  
epochs: 0% | 0/36 [00:17<?, ?it/s, loss=2.61, loss_point=0.879, lr=0.00209, d  
train: 11% | 127/1204 [00:17<02:54, 6.19it/s, total_it=127]
```

注：训练所需时间很长。推荐将课程训练好的权重数据导入，只进行测试评估。

# Part 5 | Semantic segmentation

实验流程

## » 本地测试评估 (可选)

计算机有独立显卡且显存比较大 (batch=1) , 可用使用本地测试评估:

1. 下载NAS中或github代码包, 打开pcseg32文件夹。

2. 修改/tools/cfgs/voxel/semantic\_kitti/目录下  
cylinder\_cy480\_cr5\_r1.yaml文件中

```
DATA_PATH: '/media/z/Share/study/dataset/train_data' # h  
  
EVALUATION:  
    is_save_label: True  
    save_path: /media/z/Share/study/dataset/labelpredict
```

data\_path为上一步骤代码构建的数据集的位置  
save\_path为预测标签的保存路径

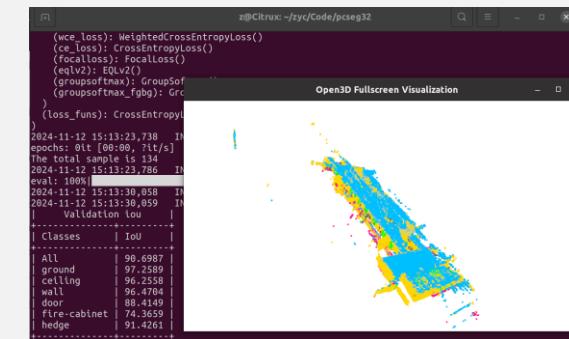
3. 将从NAS上下载的checkpoint\_epoch\_36.pth复制到  
pcseg32/logs/voxel/semantic\_kitti/cylinder\_cy480\_cr5  
\_r1/default/ckpt/目录下 (没有该目录可以运行一下本地训  
练生成)

3. 在pcseg32根目录下打开终端, (先切换pcseg虚拟环境)  
执行:

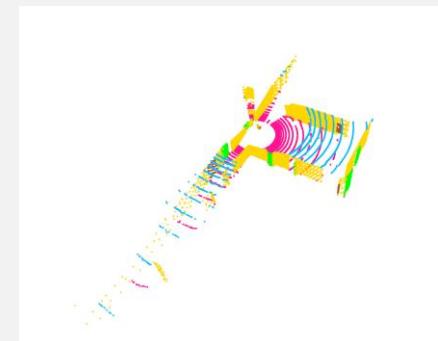
```
$ python train.py --cfg_file  
tools/cfgs/voxel/semantic_kitti/cylinder_cy480_cr5_r1.  
yaml
```

成功启动本地测试如下图所示:

多帧分割叠加结果



单帧分割结果



# Part 5 | Semantic segmentation

实验流程

## » 服务器测试评估 (可选)

计算机无独立显卡，可用使用服务器测试评估

1. 下载NAS中或github代码包，打开pcseg32文件夹。
2. 修改/tools/cfgs/voxel/semantic\_kitti/目录下 cylinder\_cy480\_cr5\_r1.yaml文件中

```
DATA_PATH: '/media/z/Share/study/dataset/train_data' # h  
EVALUATION:  
    is_save_label: True  
    save_path: /media/z/Share/study/dataset/labelpredict
```

data\_path为上一步骤代码构建的数据集的位置  
save\_path为预测标签的保存路径

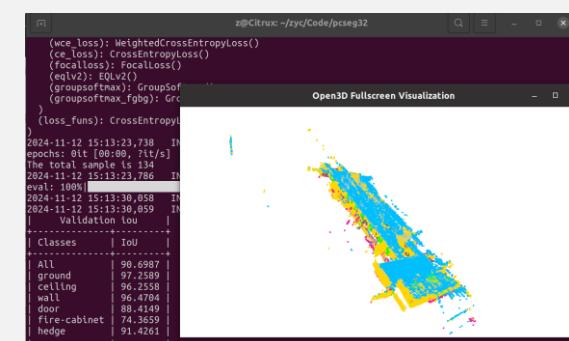
3. 将从NAS上下载的checkpoint\_epoch\_36.pth复制到 pcseg32/logs/voxel/semantic\_kitti/cylinder\_cy480\_cr5\_r1/default/ckpt/目录下（没有该目录可以运行一下本地训练生成）

3. 在pcseg32根目录下打开终端，(先切换pcseg虚拟环境)  
执行：

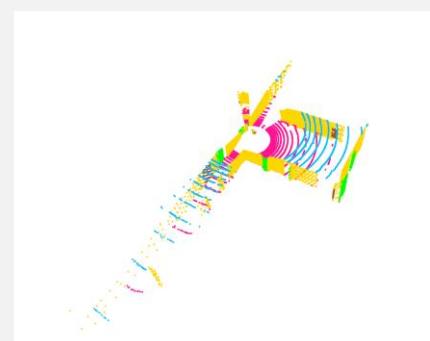
```
$ python train_jk_c.py --cfg_file  
tools/cfgs/voxel/semantic_kitti/cylinder_cy480_cr5_r1.  
yaml
```

成功启动测试评估如下图所示：

多帧分割叠加结果



单帧分割结果

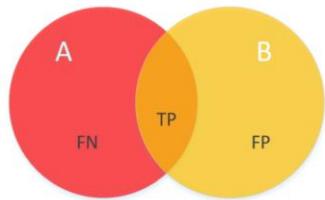


# Part 5 | Semantic segmentation

实验流程

## » IoU, mIoU评估指标, 混淆矩阵

IoU 及 mIoU



$$IoU = \frac{TP}{TP + FN + FP}$$

$$mIoU = \frac{1}{k+1} \sum_{i=0}^k \frac{TP}{FN + FP + TP}$$

混淆矩阵

图中A为真实标签, B为预测标签。

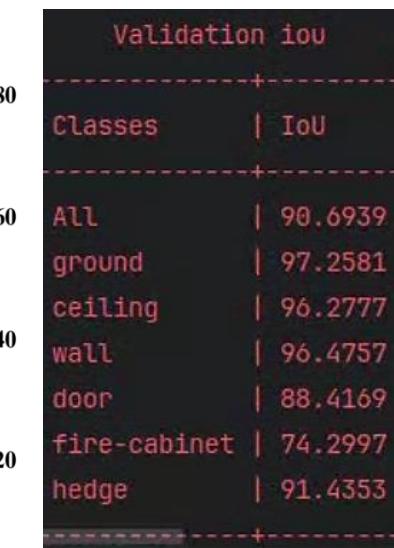
IoU就是真实标签和预测标签集合, **交集与并集的比值**。

即图中的  
 $TP / (FN + TP + FP)$   
**表征分割的精度**。

mIoU即所有标签种类的IoU的平均值

评估结果

Confusion Matrix						
True Label	Ground	Ceiling	Wall	Door	Cabinet	Hedge
Ground	98.37	0.00	1.30	0.15	0.01	0.18
Ceiling	0.01	97.57	2.34	0.04	0.00	0.03
Wall	0.37	0.25	98.48	0.60	0.21	0.09
Door	0.18	0.01	5.41	93.90	0.50	0.00
Cabinet	0.07	0.00	14.26	0.02	85.65	0.00
Hedge	2.62	0.00	3.10	0.00	0.00	94.29



真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

混淆矩阵**对角线**上即为**分割正确的百分比**, 其他部分即为错分种类, 可以反映不同标签的**错分情况**

可采用utils目录下的confusion\_matrix.py进行混淆矩阵结果的绘制

### » 实验内容

- 课程3作业（以子123文件夹的形式组织）：
  - 基于bag文件和pose.txt进行LOAM算法配置，通过回放bag生成算法估计的位姿结果，存为pose\_loam.txt。同时，利用数据存储节点，将执行过程的点云存为pcd，将位姿存为odom文件。
  - 使用EVO工具对比给出的pose.txt和本地算法结果pose\_loam.txt，绘制pose轨迹，APE和RPE。
  - 通过interactive\_slam工具，导入pcd+odom，进行手动地图关键帧选取和地图优化，保存优化后的点云地图为pcd。
  - 提交上述程序运行生成的文件，以及对比的可视化结果（红色内容）。
- 课程4作业（以子1234文件夹的形式组织）：
  - 基于签到分组分配的点云子集，进行数据标注，提交可视化标注结果的截图。
  - 在本地或者服务器远程，调用不同分割模型获取处理结果，提交模型处理label和自己标注label (GT)。
  - 基于自己标注的GT和模型处理的label进行mIoU的计算，并且绘制混淆矩阵，分析不同类别的分割结果差异，绘制不同模型的对比表格（一个）和混淆矩阵图片（按模型）。
  - 提交一个对本门课学习的总体实验心得报告（800字左右）：包括传感器使用、代码、环境配置，以及课程需要改进的内容（word形式）。

# 开始实验

<https://github.com/hahakid/3D-point-cloud-processing-and-visualization-practice/tree/main/lecture4>



北京航空航天大学  
BEIHANG UNIVERSITY