



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)

Lecture3-激光雷达SLAM&交互SLAM

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：贾皇城、梁瞳尧



邮箱：ouyangkid@buaa.edu.cn



学期：2024年秋季

目录

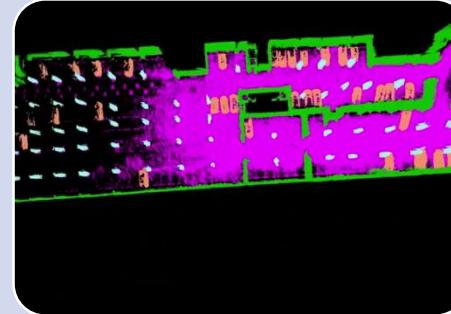
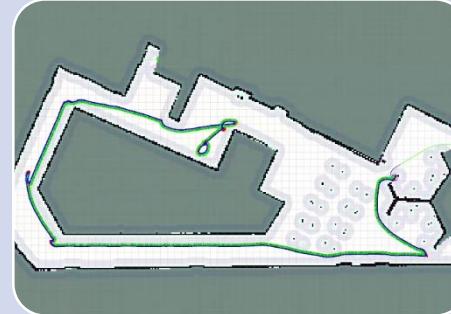
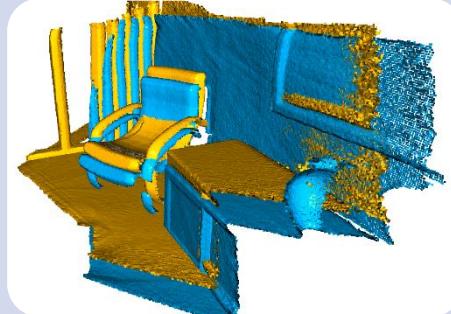
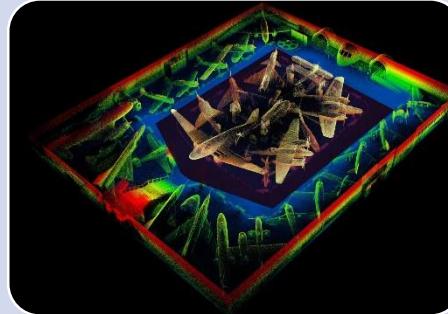
Contents

- 01 课程内容
- 02 课程目标
- 03 学习目标
- 04 实验设备及数据
- 05 SLAM&交互SLAM

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

» 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

» 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

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 SLAM&交互SLAM

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 SLAM&交互SLAM

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
- 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2作业（以子1234文件夹的形式组织）：
 - 基于0_prepare_data.py 进行目标环绕拍摄，生成pcd文件，并基于cc进行裁剪。
 - 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？旋转改为平移，以及旋转+平移。
 - 在Chat-GPT的引导下，编写一个kd-tree建树程序，并进行树结构打印。
 - 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。
 - 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。

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）

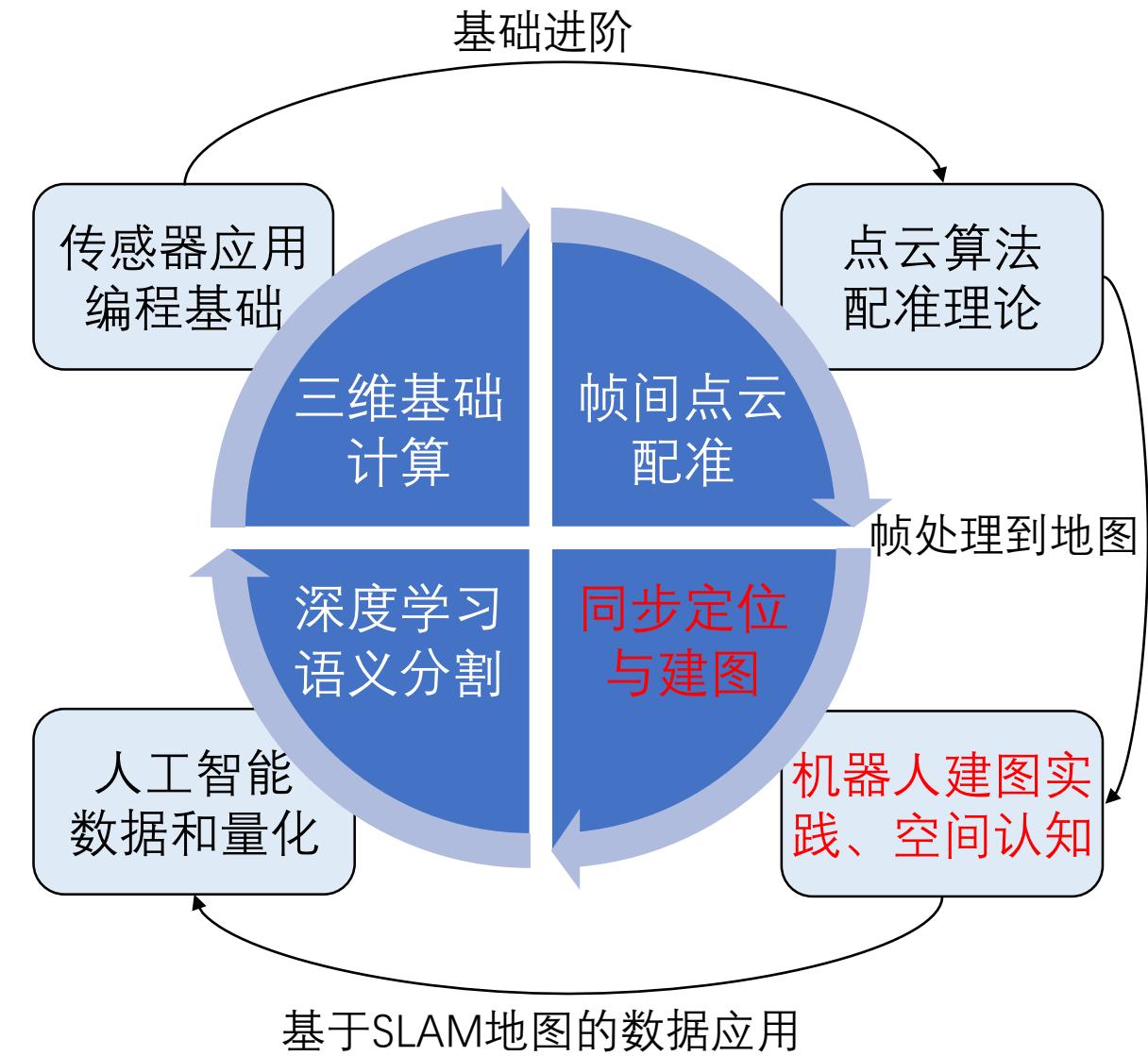
- Realsense：双目+结构光，稠密视场点云
- LiDAR：机械ToF测距，稀疏环视点云

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）

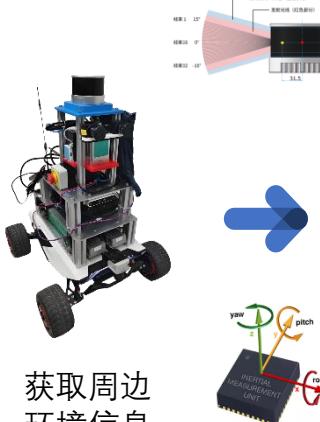
- Ubuntu: shell
- ROS机器人操作系统
- PointLabeler



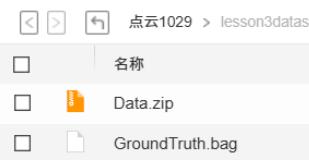
Part 3 | 学习目标

3-4课整体架构

» 3D点云数据处理和可视化实践入门

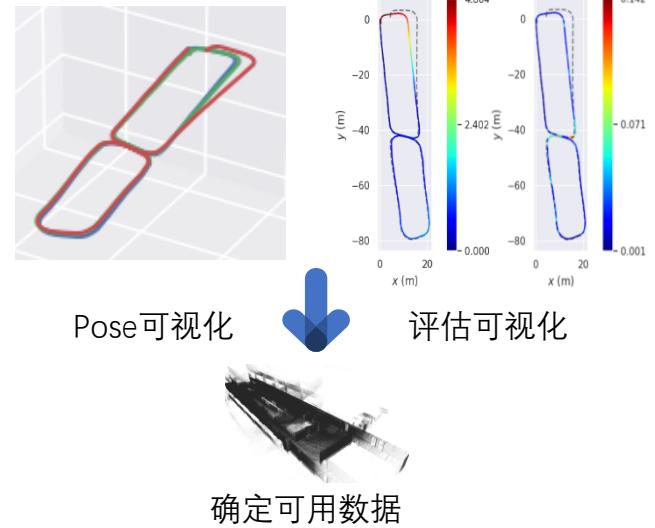
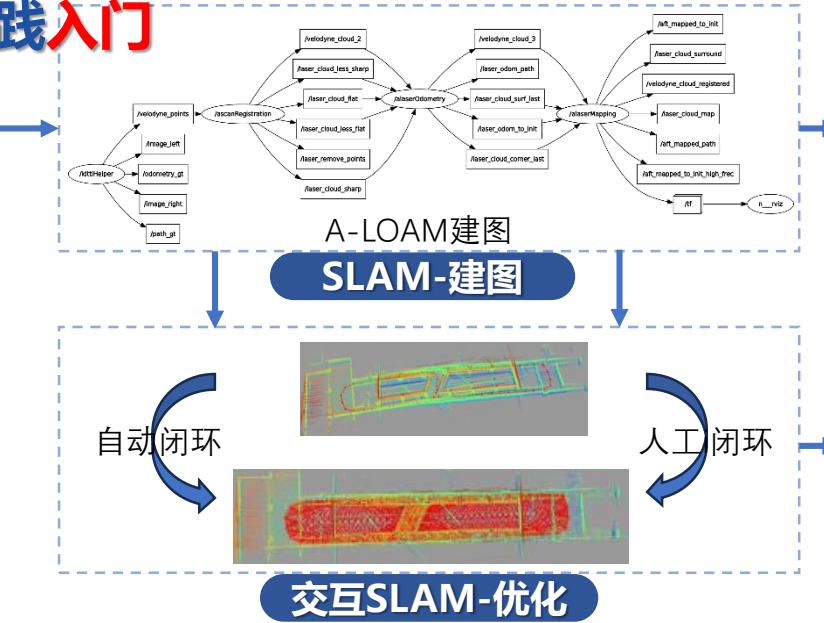


获取周边环境信息



bag数据保存于NAS

数据准备



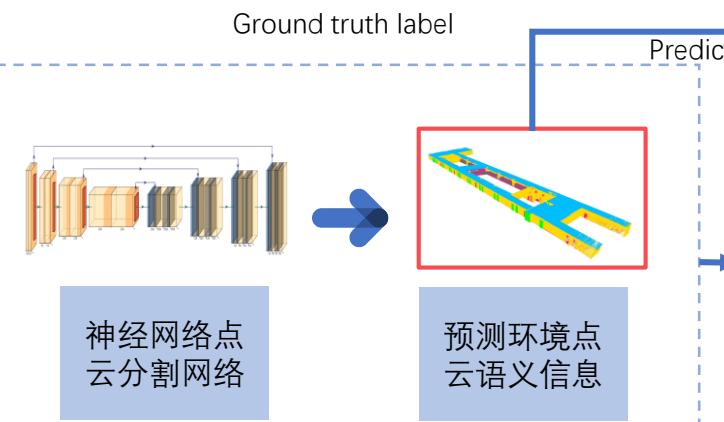
评估



接收kitti
格式数据

Point
Labeler
数据标注

赋予语义
标签



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

各label种类的分割精度

数据准备

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 SLAM&交互SLAM

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

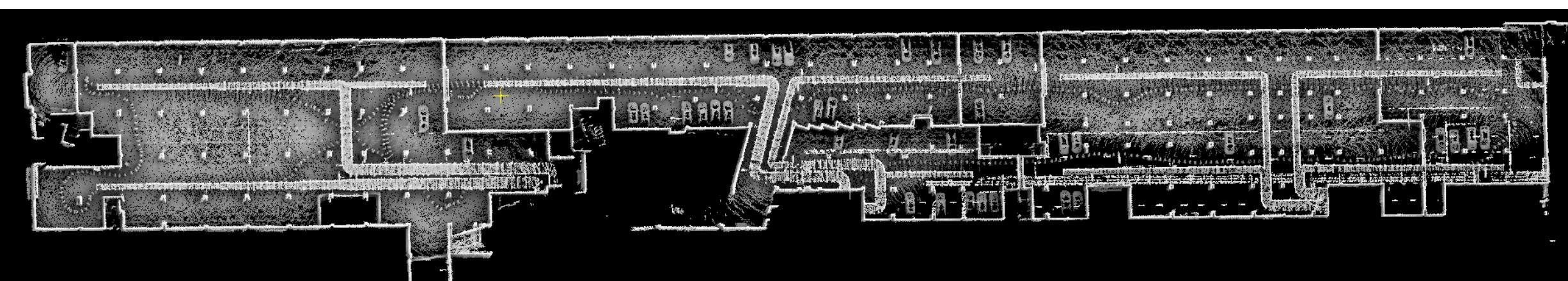
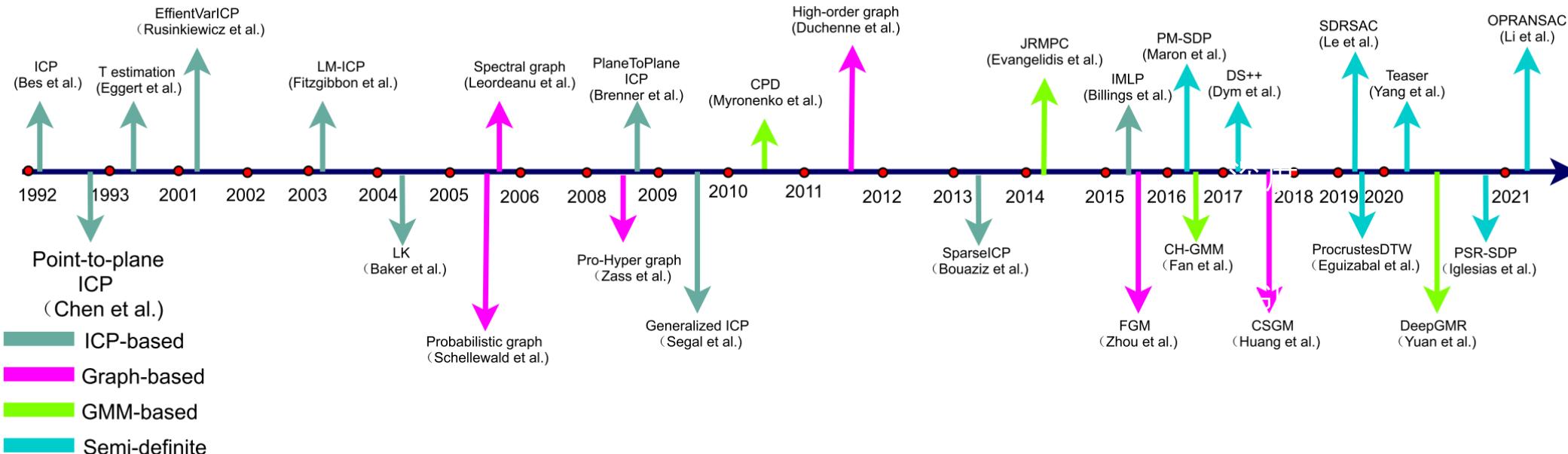
Contents

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

Part 5 | 激光SLAM

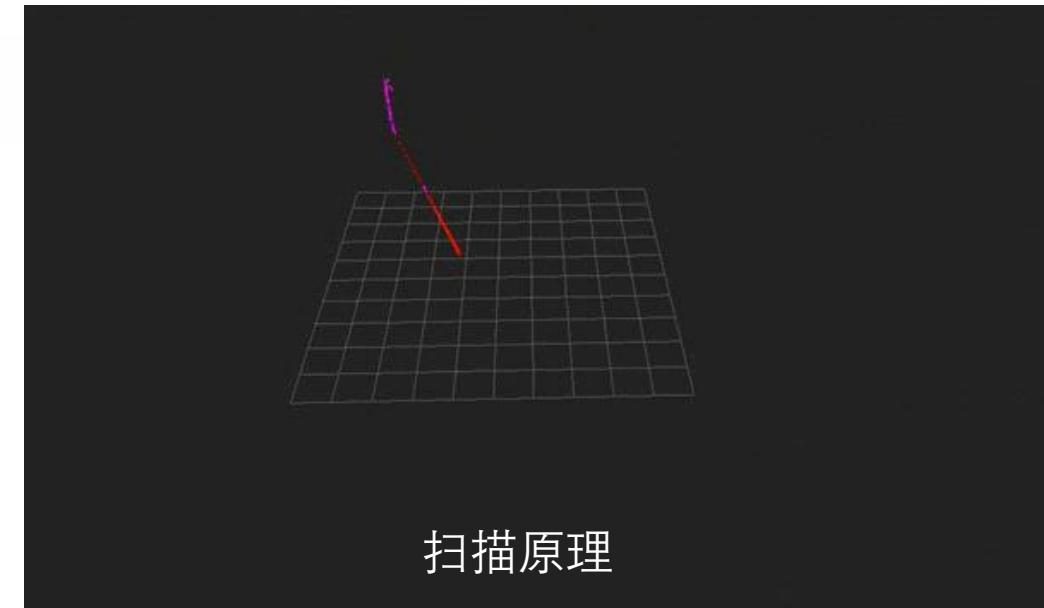
SLAM简介

» 激光雷达



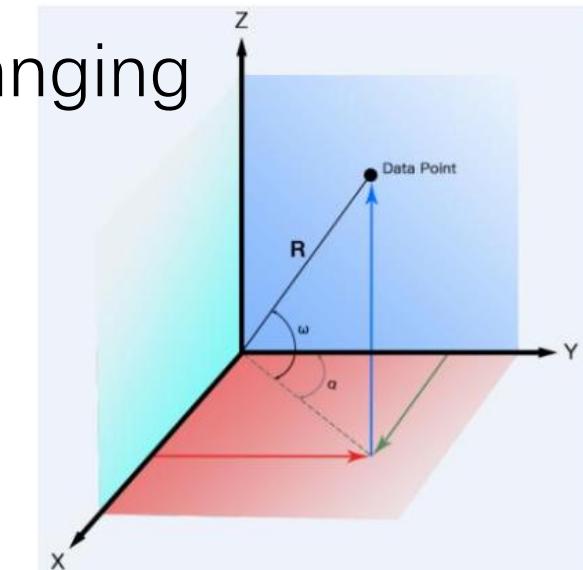
Part 5 | 激光SLAM

» 激光雷达LIDAR

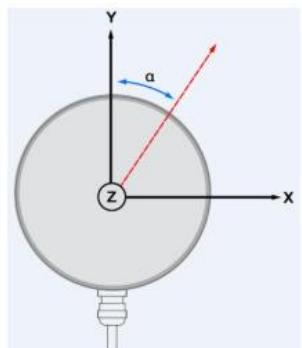


光激光探测与测距 Light Laser Detection and Ranging

- 非可见光波段: 903-908nm 波段(eye safe)
- 主动感知: 基于ToF (time of fly) 收发测距
- 精准测距: 空间三维信息+反射率: xyzr
- 环视: 水平FOV=360°
- 数据: 稀疏、不规则、低频(5-20Hz)、运动畸变敏感

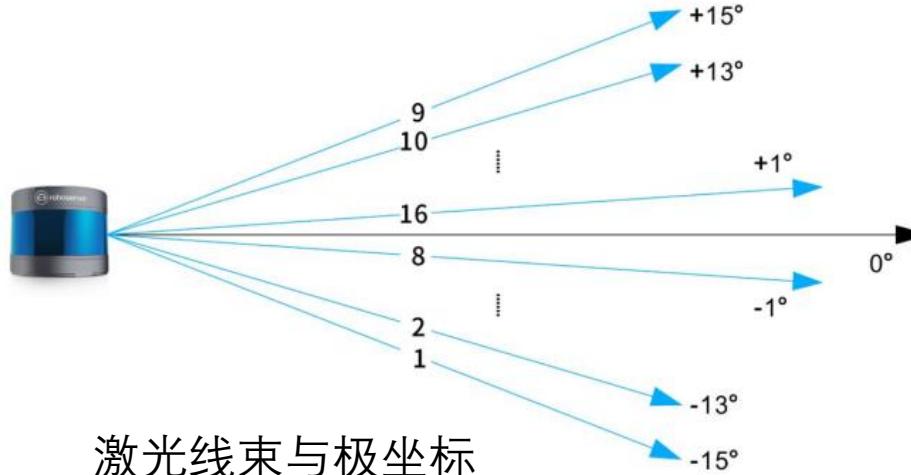


坐标系



Part 5 | 激光SLAM

» 激光雷达



激光线束与极坐标

$$\begin{aligned}x &= r \cos(\omega) \cos(\alpha); \\y &= r \cos(\omega) \sin(\alpha); \\z &= r \sin(\omega);\end{aligned}$$

↓

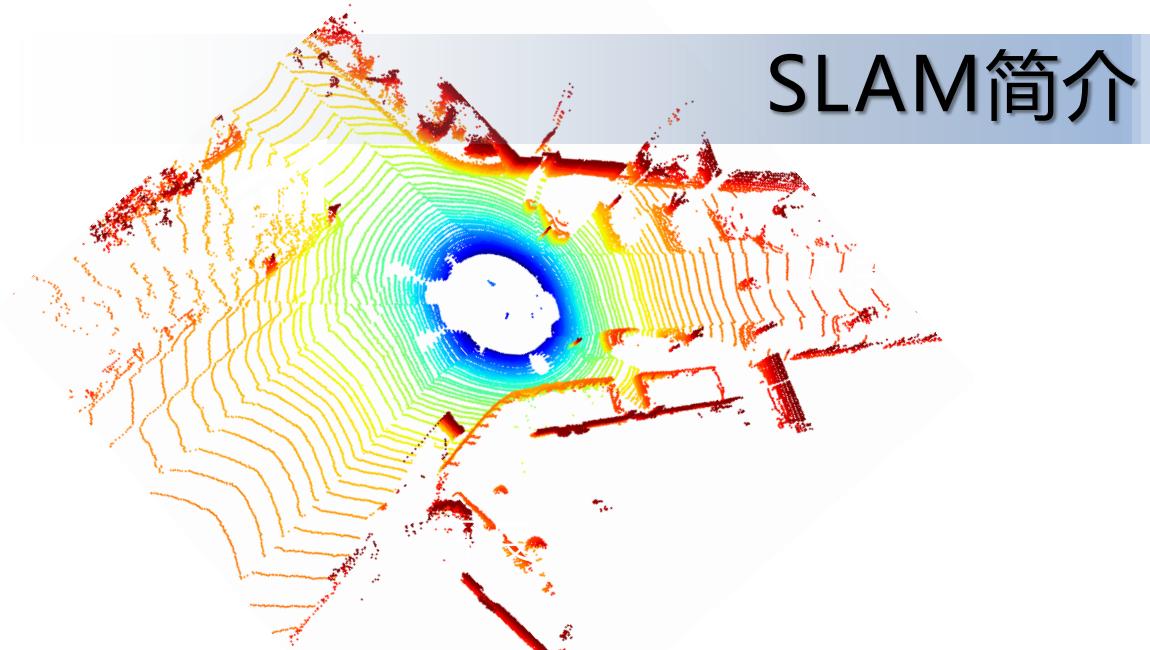
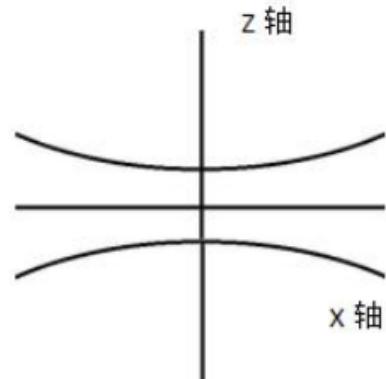
$$x^2 + y^2 = z^2 \cos^2(\omega) / \sin^2(\omega)$$

↓

$$\frac{z^2}{(y \tan(\omega))^2} - \frac{x^2}{y^2} = 1$$

16线

A	B
1	-15
2	-13
3	-11
4	-9
5	-7
6	-5
7	-3
8	-1
9	15
10	13
11	11
12	9
13	7
14	5
15	3
16	1



- 激光雷达扫描线, `scan_id/ring number`, 在出厂前通过预标定角分辨率和张角度数确定。
- 能够提供重要的空间几何信息, 最新版本的雷达会直接在SDK提供该信息。
- `<scan_id, w>`也在一些点云投影算法中被广泛使用, 通过左边公式可进行欧氏空间转换。

SLAM简介

» 激光雷达 vs. 双目深度相机

Stereo/depth camera:

- 预标定结构光/光栅，视差测距
- 主要用于室内近距离场景
- 密度大
- 视野小、盲区大(基线影响)
- 适用天气广泛
- 同类传感器互相不影响
- 受环境光影响
- 短距离测距(<1m)误差大
- 远距离精度衰减严重
- 提供纹理



在户外、大型场景的SLAM
领域主要以LiDAR为主。

LiDAR

- 主动激光扫描线往返时间
- 适用室内外环境 ←
- 稀疏
- 环视视野大 ←
- 受雨雾灰尘影响
- 同类设备互相干扰 (波段)
- 不受环境光影响 ←
- 长距离测距，精准 ←
- 仅提供反射率 (缺少纹理)

Part 5 | 激光SLAM

传感器介绍

激光雷达

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）

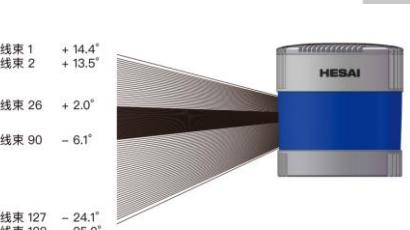


图 1.5 线束分布示意图

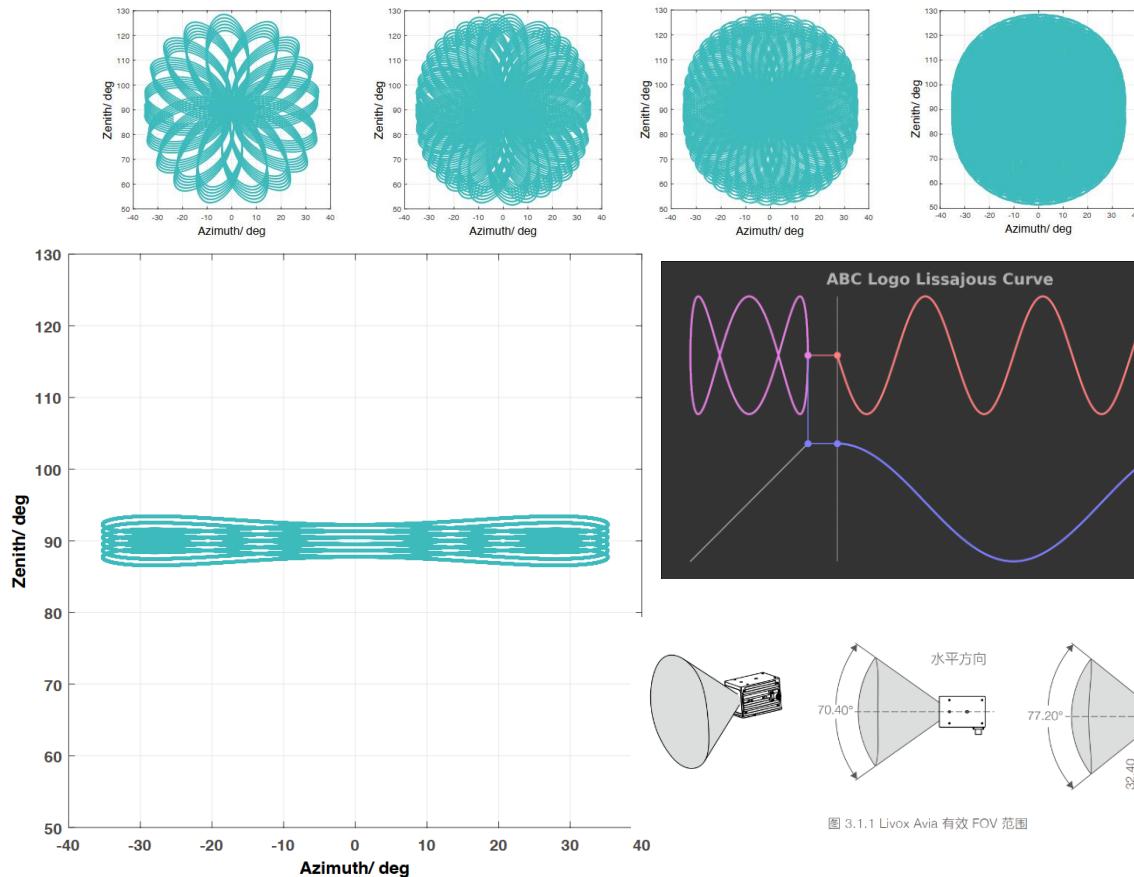
Part 5 | 激光SLAM

传感器介绍

» 激光雷达

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）



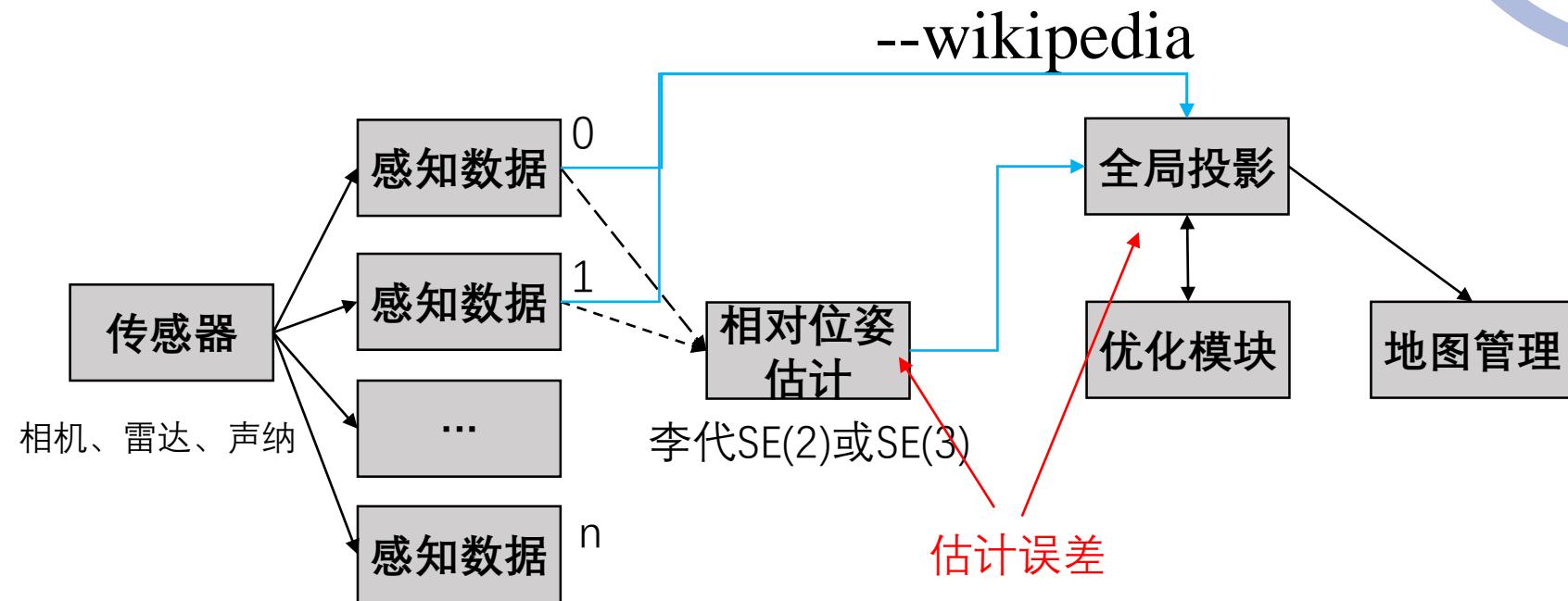
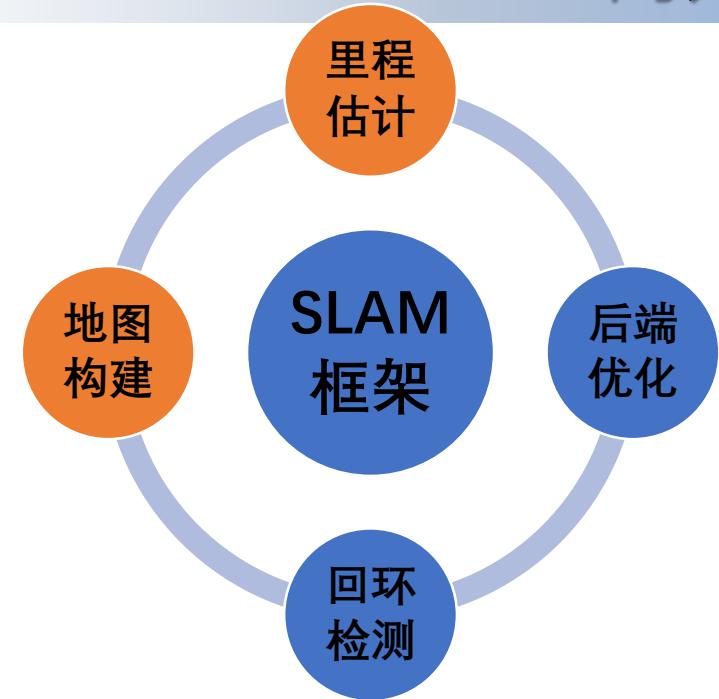
产品型号	MID-40	HAP	TELE-15	MID-70	AVIA
应用场景	移动机器人、智慧物流等	自动驾驶、智能辅助驾驶系统等	轨道交通、智慧交通	无人配送车、园区物流车等	测绘、安防、智慧城市等
量程 ¹ (@ 100 klx)	90 m @ 10% 反射率 130 m @ 20% 反射率 260 m @ 80% 反射率	150 m @ 10% 反射率 200 m @ 80% 反射率	320 m @ 10% 反射率 500 m @ 50% 反射率	90 m @ 10% 反射率 130 m @ 20% 反射率 260 m @ 80% 反射率	190 m @ 10% 反射率 230 m @ 20% 反射率 320 m @ 80% 反射率 450 m @ 80% 反射率, 0 klx
虚警率 ² (@ 100 klx)	< 0.01%	< 0.01%	< 0.01%	< 0.01%	< 0.0003%
距离精度 ³ (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

Part 5 | 激光SLAM

SLAM简介

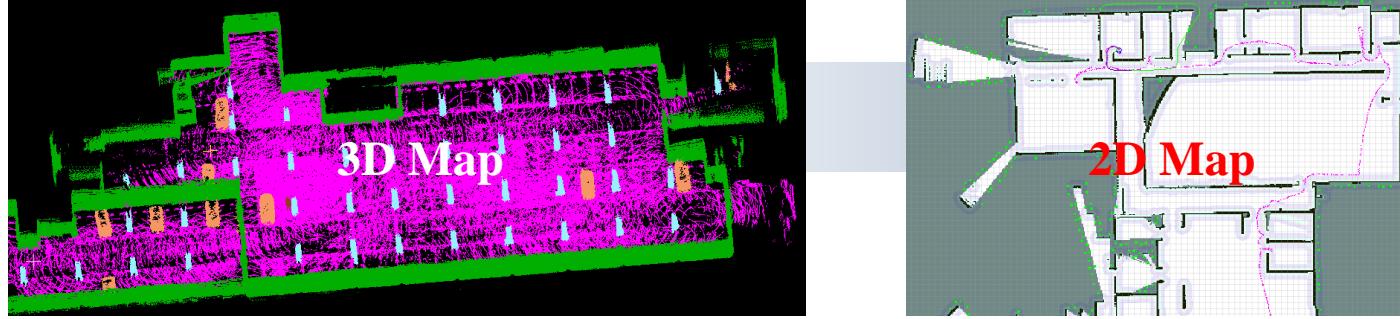
» 激光雷达

Simultaneous localization and mapping (SLAM) is the computational problem of **constructing or updating a map of an unknown environment** while simultaneously keeping track of an agent's location within it.



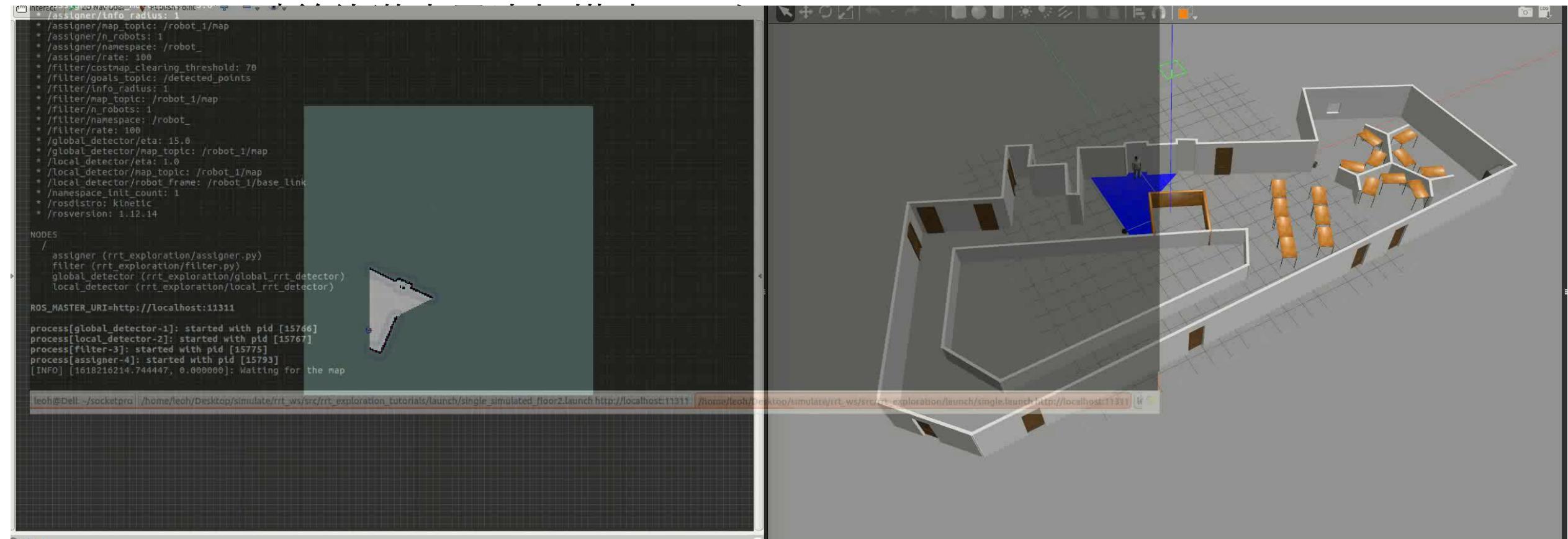
Part 5 | 激光SLAM

» 激光雷达SLAM



- 2D SLAM:

SE(2)



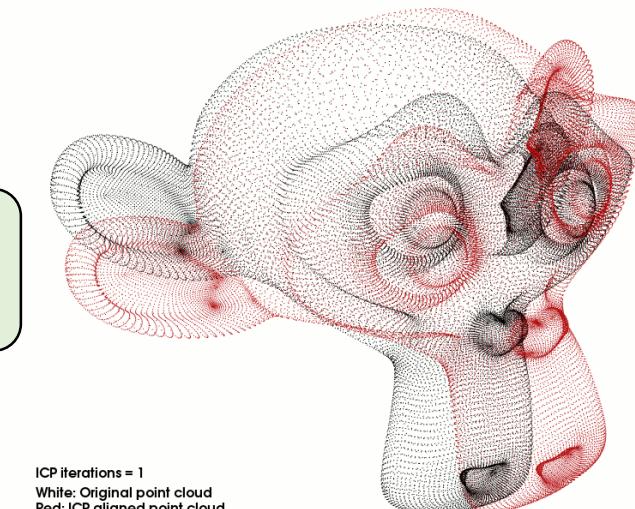
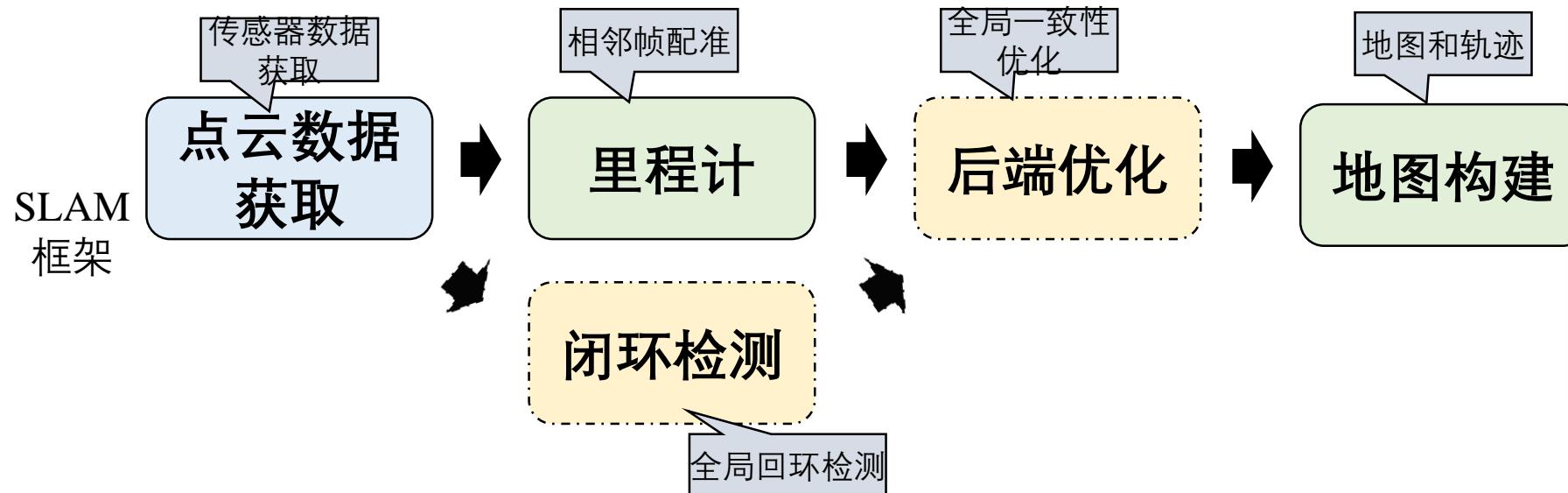
- 具备语义和可解释性扩展可能。

Part 5 | 激光SLAM

SLAM简介

» SLAM

- **3D SLAM直接法**: 全局ICP、NDT等，缺点是全局匹配耗时、计算复杂、无法满足实时性要求。
- **3D SLAM间接法**: 通过寻找几何特征稳定、可区分度高的特征，然后基于特征相关性进行配准，结合不同优化方法，能够满足实时性需求。

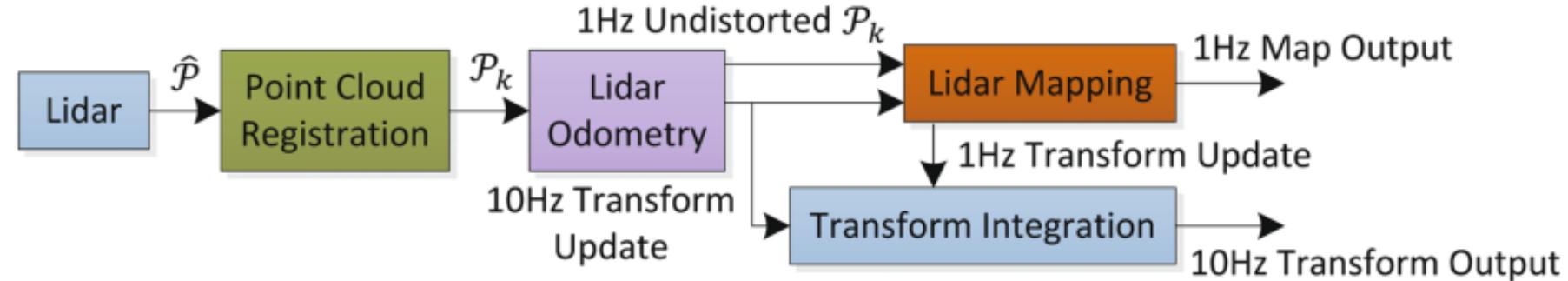
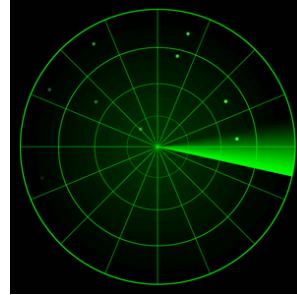


Part 5 | 激光SLAM

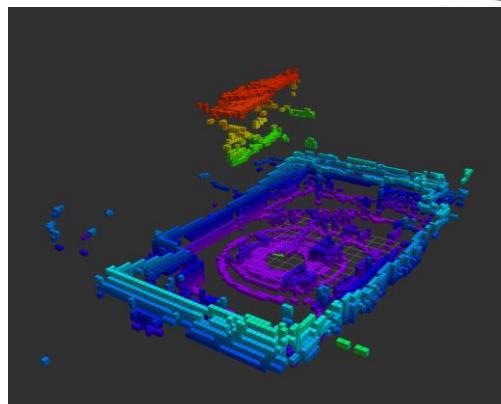
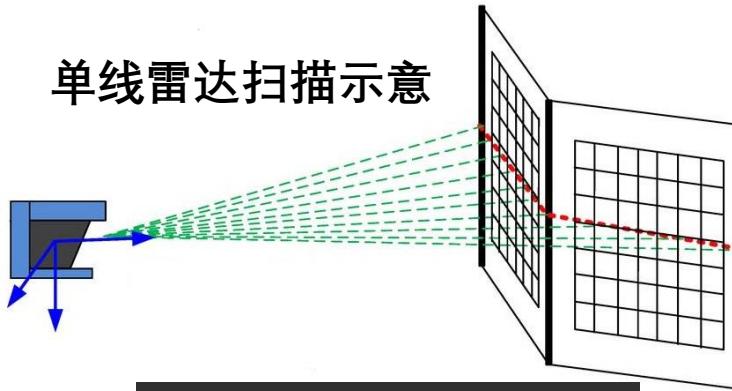
LOAM



LOAM: Low-drift and real-time lidar odometry and mapping(2014)



单线雷达扫描示意



- **传感器方案：**针对单线激光和2D SLAM的缺陷，作者设计了步进电机+单线雷达组合实现三维扫描的方案。(最近几年随着无人驾驶技术的发展，通过内置偏光镜的多线激光雷达成本降低到万元以内)
- **SLAM算法：** LOAM适配雷达点云的低漂移、实时雷达里程计与建图算法 - LOAM。
- 后续研究人员对算法改进，适配了现有的多线雷达，并在此基础上进行了各种优化和扩展。

Part 5 | 激光SLAM

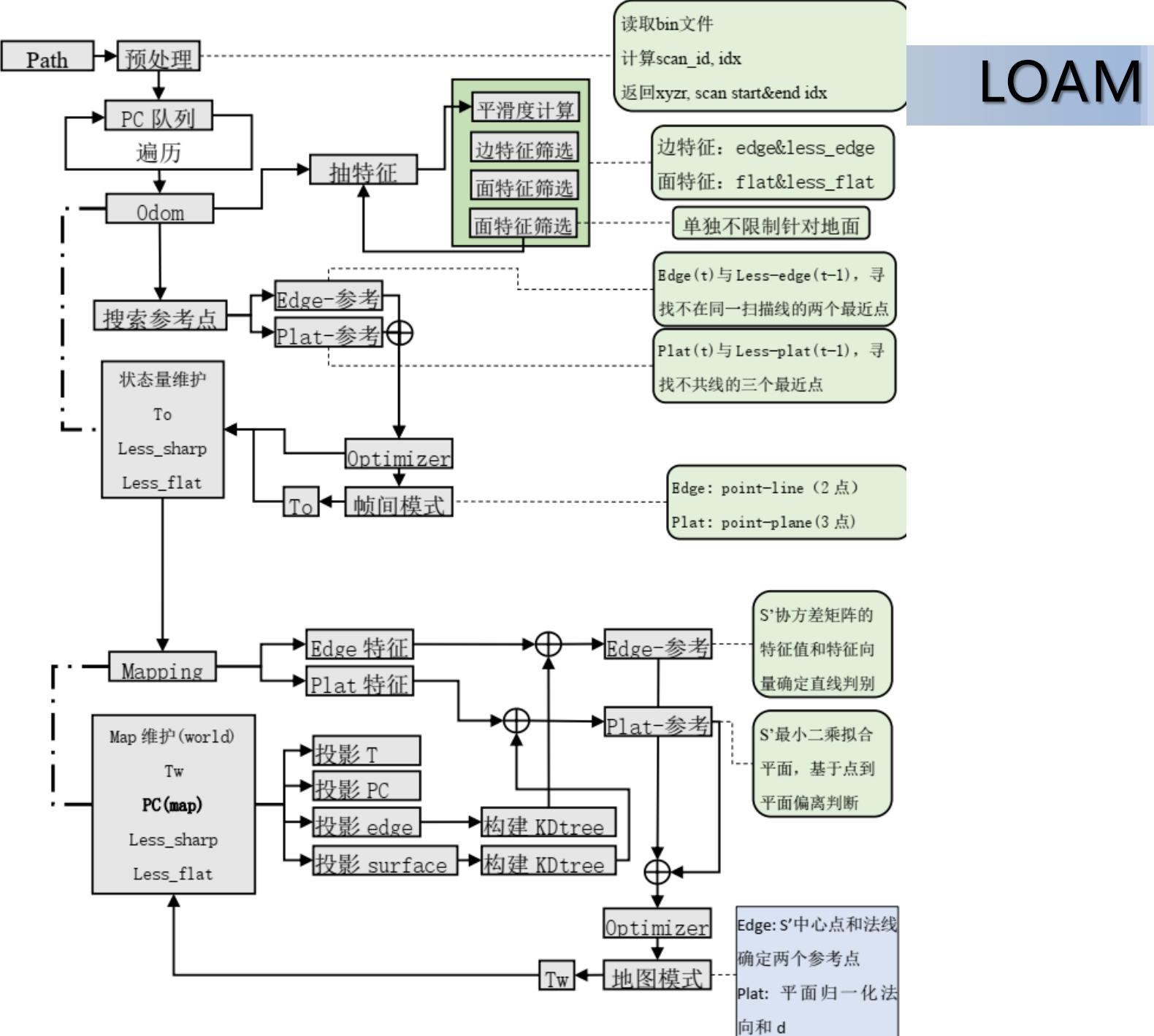
» LOAM

• Odometry

1. 特征提取
2. 关联特征搜索
3. 里程计算算法

• Mapping

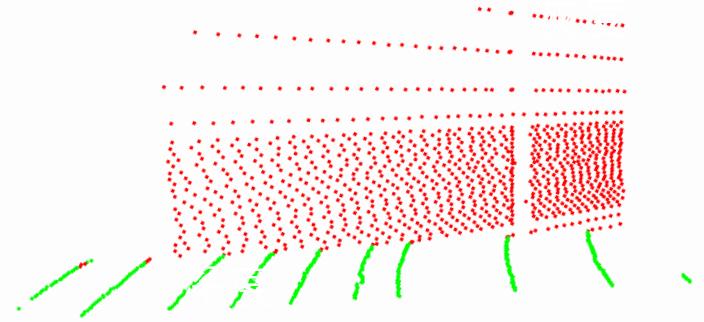
1. 特征提取
2. 地图匹配
3. 降采样



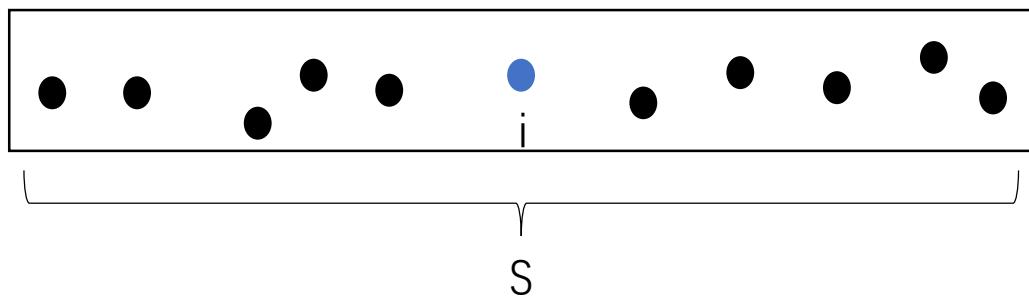
» LOAM

- 平滑度公式

$$c = \frac{1}{|\mathcal{S}| \cdot \|X_{(k,i)}^L\|} \left\| \sum_{j \in \mathcal{S}, j \neq i} (X_{(k,i)}^L - X_{(k,j)}^L) \right\|.$$



基于雷达激光束扫描过程的连续性，通过一个滑动窗口对点云进行遍历，然后基于点云的空间坐标距离波动来估计点的平滑程度，从而确定点属于平面Planer ($c < \text{threshold}$)，还是边缘Edge ($c > \text{threshold}$)。



当前点与窗口内点差异**越大**，属于**边缘**；
当前点与窗口内点差异**越小**，属于**平面**；

Part 5 | 激光SLAM

» LOAM-特征提取

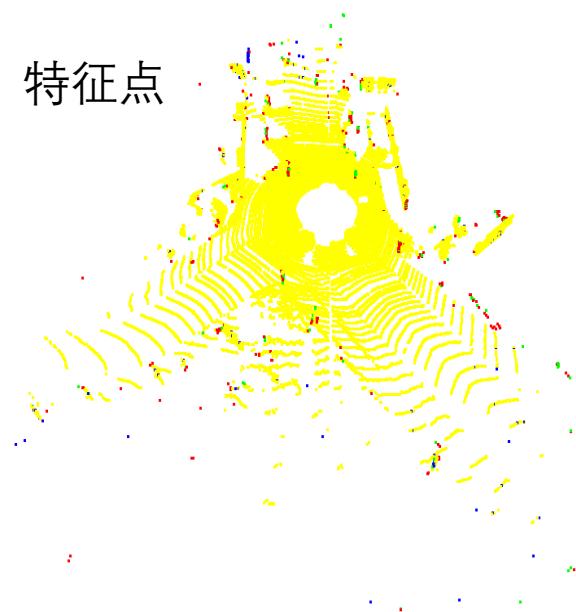
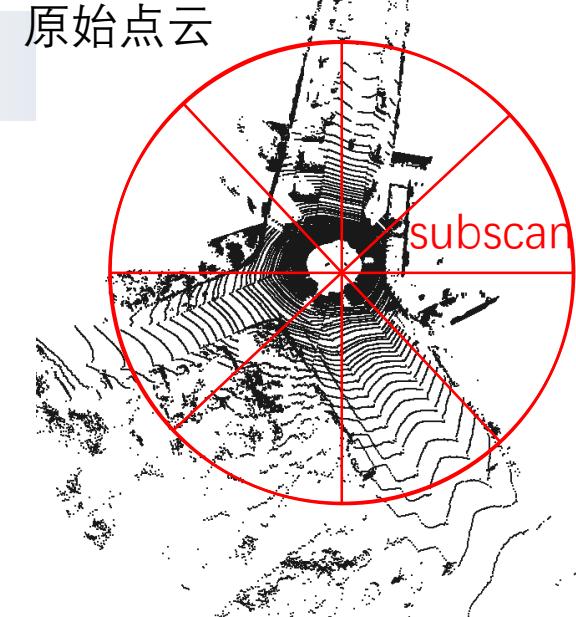
- 对多线激光雷达扫描时，可以基于雷达的Scan_id进行顺序遍历；基于yaw进行等份切分。
- 并且基于平滑度排序确定edge和less_edge;flat和less_flat。

提升特征点质量

特征点选择依据：

- 两类特征基于平滑度公式区分，且具有判断的阈值条件（**特征可区分性**）
- 通过将每个扫描线进行等间距划分，使得特征空间分布均匀（**特征空间分布均匀**）
- 通过限制每个子区域的总特征数量，保证计算效率(限制特征总体数量)
- 通过控制相邻特征点距离，防止特征点过于集中（**特征空间分布均匀**）
- 避免与扫描线束垂直的平面特征被选中，通过平面法向夹角进行限制($>10^\circ$)（**特征稳定性**）
- 避免选择被遮挡/阻塞区域的边界特征点，方式视角转换后特征发生变化（**特征稳定性**）

	edge	less_edge	flat	less_flat
理论上限	1536	7680	1536	未设限
实际1	235	268	336	83012
实际2	252	284	336	84928



» LOAM-关联特征搜索

全局ICP问题：

- 环境动态性和点云随机噪声使得相邻数据帧计算的点云无法一一匹配；
- 配准时进行暴力搜索和遍历的效率低下。

针对上述问题：

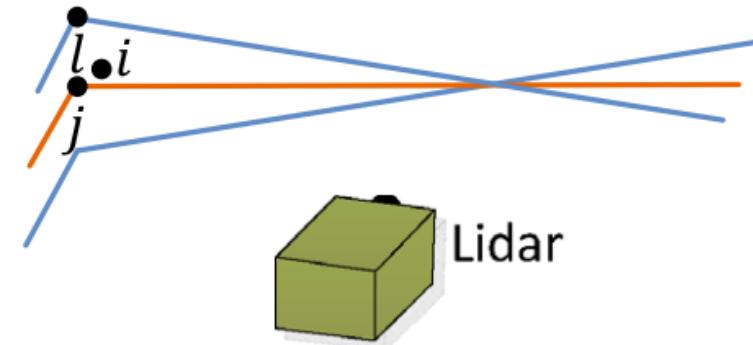
1. 引入KD-Tree协助搜索关联特征：给定 t 时刻的点云 $P(t)$ 的特征 $\text{Edge}(t)$ 和 $\text{Flat}(t)$, 在 $t-1$ 时刻点云 $P(t-1)$ 的 $\text{Less_edge}(t-1)$ 和 $\text{Less_flat}(t-1)$ 分别进行近邻搜索。
2. 关联特征为近邻点附近的点，针对边特征构建“点-线”配准的关联特征；针对面特征构建“点-面”配准的关联特征。

» LOAM-关联特征

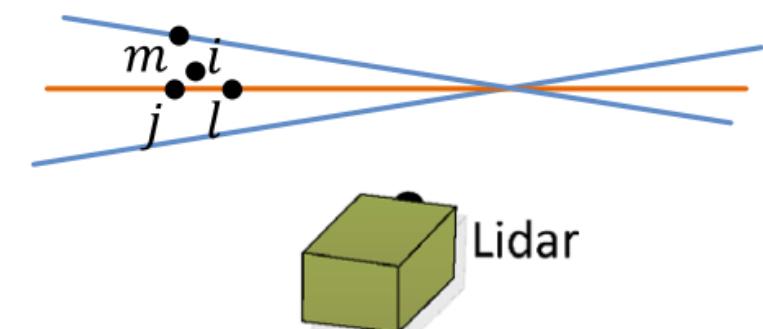
Odometry的点云配准策略：

- (a) 针对边特征 i , 寻找 $t-1$ 时刻边特征中最近的点 j , 以及相邻扫描线上的次近点 l 。 j 和 l 构成直线, 求解 i 到对应直线的距离。
- (b) 针对面特征 i , 寻找 $t-1$ 时刻面特征中的最近的点 j , 与 j 共扫描线的次近点 l , 以及与 j 不同扫描线的次近点 m 。 j , l 和 m 保证了不共线, 形成一个平面, 求解 i 到对应平面的距离。
- 通过寻找多组(a)和(b)约束特征组, 基于最小二乘法求解使得最优的距离约束, 得到一组相邻点云的相对位置关系, 以及由 $P(t-1)$ 到 $P(t)$ 的旋转平移矩阵 $T \in SE(3)$.

*在使用多线激光雷达时, 相邻扫描线为平行。



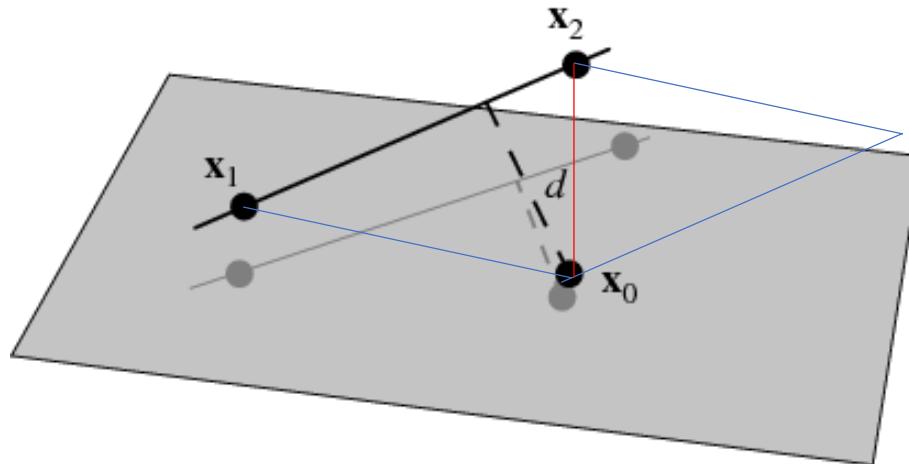
(a)



(b)

» LOAM-特征提取

- Edge的关联特征-点线距离



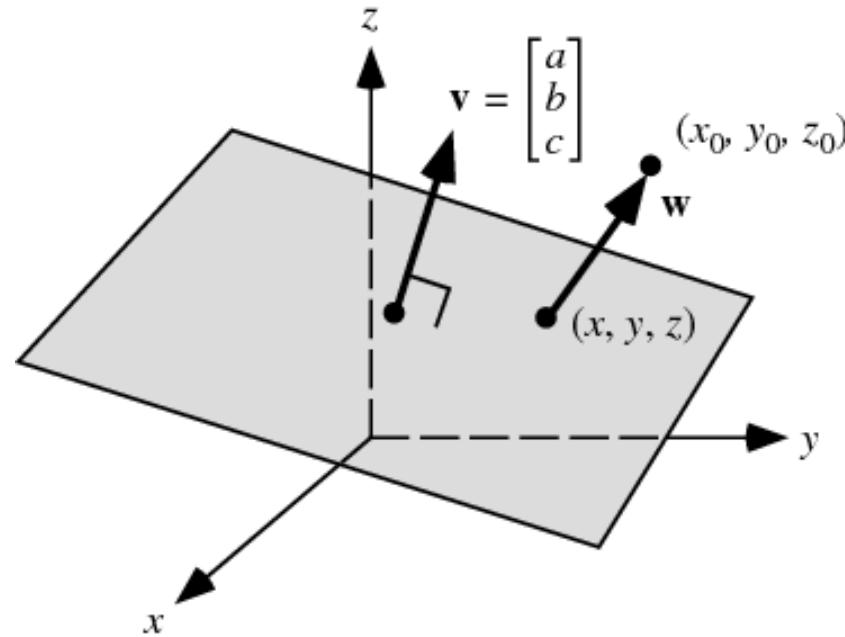
已知点 x_0 和目标直线上的两个点 x_1 和 x_2 ，基于线性代数，可以基于公式求的距离d。

$$\begin{aligned} d &= \frac{|(x_2 - x_1) \times (x_1 - x_0)|}{|x_2 - x_1|} \\ &= \frac{|(x_0 - x_1) \times (x_0 - x_2)|}{|x_2 - x_1|} \end{aligned}$$

平行四边形面积=2×三角形面积=2×(1/2×|x₂-x₁|×d)

» LOAM-特征提取

- Planer的关联特征-点面距离



已知点 x_0 和目标平面上的三个点 x_1 ， x_2 和 x_3 ，
基于线性代数，先求由三个点构成平面的法
向量。

$$\hat{\mathbf{n}} = \frac{(x_2 - x_1) \times (x_3 - x_1)}{|(x_2 - x_1) \times (x_3 - x_1)|}$$

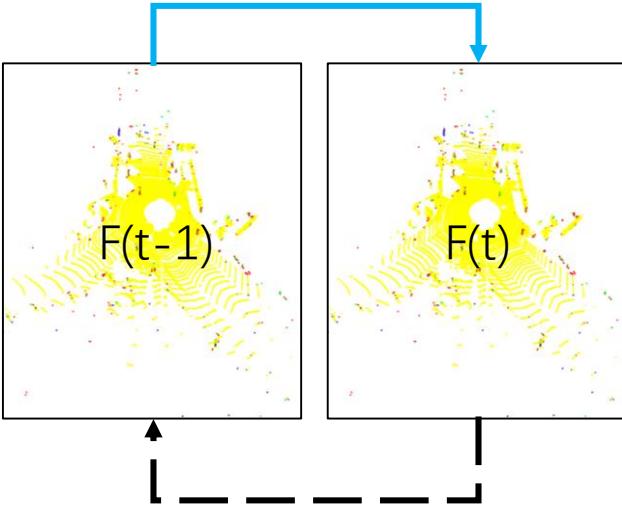
然后基于平面任意一点和法向，可得距离为

$$D_i = \hat{\mathbf{n}} \cdot (x_0 - x_i), i = 1, 2, 3$$

*第二个公式为向量点乘

» LOAM-位姿估计

现实：机器人运动，传感器位姿变化



配准：基于优化方法求解 $\Delta = F(t) - F(t')$

求解目标

- 户外场景下的SLAM位置估计一般假设是三维刚体运动，包含平移变换

$$t=(x,y,z), \text{ 姿态旋转 } R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad T = (R, t) = \begin{bmatrix} & & x \\ & R & y \\ 0 & 0 & z \\ 0 & 0 & 1 \end{bmatrix}$$

- 设初始0时刻 $T_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, 通过矩阵乘法可以将每帧点云位姿矩阵投影到全局坐标上。

- 基于所获得的边特征和关联特征，面特征和关联特征分别计算得到多组距离约束 $D=\{d_1, d_2, \dots, d_n\}$, 基于最小二乘拟合T矩阵即可。为了方便计算，一般将T矩阵转化为六元组进行计算 $T=[x, y, z, roll, pitch, yaw]$ 。基于欧拉角公式，对传感器在三个坐标轴的旋转角代替R矩阵。（*该步骤基于现有优化工具求解即可）

» LOAM-最小二乘

$$\text{目标函数} = \sum (\text{观测值} - \text{理论值})^2$$

目标函数：使得基于T投影后，两帧（特征）点云F(t-1)到F(t)之间的距离最小。

观测值：基于边/面特征点和关联特征点求解的“点-线”、“点-面”距离，初始位姿矩阵T(t-1)。

理论值：拟合函数(高次多项式)。

$$h_{\theta}(x) = \sum_{i=0}^n \theta_i x^i$$

目标函数：

- 点线距离目标函数
- 点面距离目标函数

$$d_e = \frac{|(x_2 - x_1) \times (x_1 - x_0)|}{|x_2 - x_1|}$$

$$d_p = \frac{(x_2 - x_1) \times (x_3 - x_1)}{|(x_2 - x_1) \times (x_3 - x_1)|} \cdot (x_0 - x_1)$$

使目标函数最小化时的拟合位姿变换函数

$T = \text{LeastSqaure}(\{d_e, d_p\})$ 使得残差平方和最小

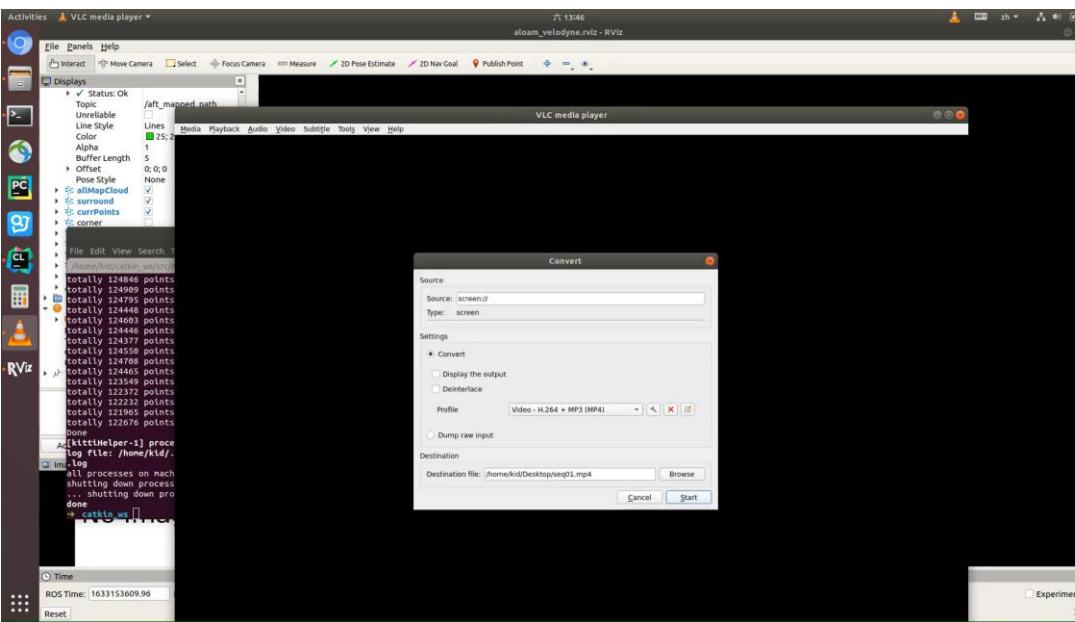
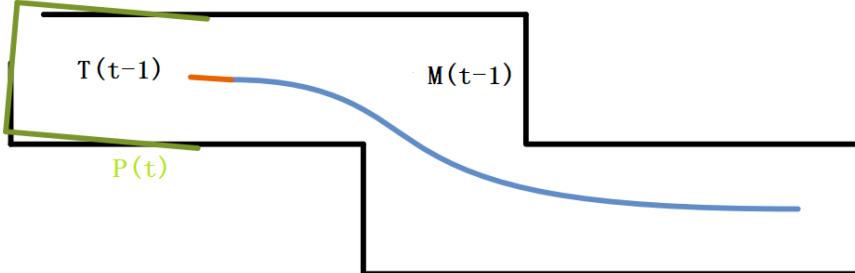
*大多数编程语言都提供了相关优化库，如sklearn, scipy, numpy.linalg.lstsq。

Part 5 | 激光SLAM

LOAM

» LOAM-建图

- 建图逻辑与相邻帧点云配准类似
- 基于t时T(t-1), 将最新点云P(t)投影道全局地图Map(t-1)上。
- 基于P(t)的特征点, 在Map(t-1)上进行搜索, 构建边特征和面特征, 以及对应的关联特征, 同样采用最小二乘求解。



橙色为里程计
估计位姿
绿色为建图估
计位姿

*重新计算特征关联主要考虑地图特征更丰富和稳定, 因此计算的里程计T序列的累计误差和最终地图偏差均较小。

» LOAM-建图

关联特征点基于已构建得到的子地图，
可用稳定参考点更加丰富。

特征点选择优化策略：

- 增加关联特征点选择数量：每个边特征(>2)和面特征(>3)都寻找点进行关联，优化关联目标。
 - 直线：求n个点中心和法向，基于固定长度求取拟合直线的固定长度端点；
 - 平面：基于n个点进行最小二乘拟合，求平面法向。
- 引入除距离判断之外的拒绝条件 (rejection condition)，排除不可靠点。
 - 直线：n个点的协方差矩阵的特征值，一大两小。
 - 平面：n个点到拟合平面的距离小于阈值(0.2m)。
- 对平面直接进行平面拟合过程求解的归一化法向和尺度，加速后续计算。

- 部署系统: Ubuntu18
- A-LOAM链接: <https://github.com/HKUST-Aerial-Robotics/A-LOAM>
- 按照A-LOAM的Github部署教程部署到执行catkin_make前
- 关键库版本: cmake=3.29.8, ceres solver=2.1.0, pcl=1.8.1

- 先修改代码:
 - CMakeLists.txt: c++11->c++14
 - Cpp
 - /camera_init->camera_init
 - /camera->camera
 - /map->map
 - kittiHelper.cpp
 - CV_LOAD_IMAGE_GRAYSCALE->cv::IMREAD_GRAYSCALE
- 执行catkin_make
- 运行
 - KITTI
 - ROSbagfile

2. Build A-LOAM

Clone the repository and catkin_make:

```
cd ~/catkin_ws/src  
git clone https://github.com/HKUST-Aerial-Robotics/A-LOAM.git  
cd ../  
catkin_make  
source ~/catkin_ws/devel/setup.bash
```

- 部署系统: Ubuntu18
- A-LOAM链接: <https://github.com/HKUST-Aerial-Robotics/A-LOAM>
- 按照A-LOAM的Github部署教程部署到执行catkin_make前
- 关键库版本: cmake=3.29.8, ceres solver=2.1.0, pcl=1.8.1

2. Build A-LOAM

Clone the repository and catkin_make:

```
cd ~/catkin_ws/src  
git clone https://github.com/HKUST-Aerial-Robotics/A-LOAM.git  
cd ../  
catkin_make  
source ~/catkin_ws/devel/setup.bash
```

- 使用KITTI数据集
 - 在kitti_helper.launch配置数据集路径, to_bag设置依照自己, 默认false
 - 注意数据集文件夹层级
 - source devel/setup.bash每次打开新的terminal都要执行一次
 - roslaunch aloam_velodyne aloam_velodyne_HDL_64.launch
 - 配置完kitti_helper中文件路径后运行roslaunch aloam_velodyne kitti_helper
- 使用rosbag
 - source devel/setup.bash每次打开新的terminal都要执行一次
 - roslaunch aloam_velodyne aloam_velodyne_HDL_32.launch脚本依据雷达而定
 - rosbag play xxx.bag

- 数据集文件夹层级
 - results---data_odometry_poses
 - txt
 - sequences---data_odometry_gray
 - 00
 - image_0
 - ...
 - 01
 - velodyne---data_odometry_velodyne
 - sequences
 - 00
 - velodyne
 - bin
 - 01

- 部署系统: Ubuntu18
- A-LOAM链接: <https://github.com/HKUST-Aerial-Robotics/A-LOAM>
- 按照A-LOAM的Github部署教程部署到执行catkin_make前
- 关键库版本: cmake=3.29.8, ceres solver=2.1.0, pcl=1.8.1

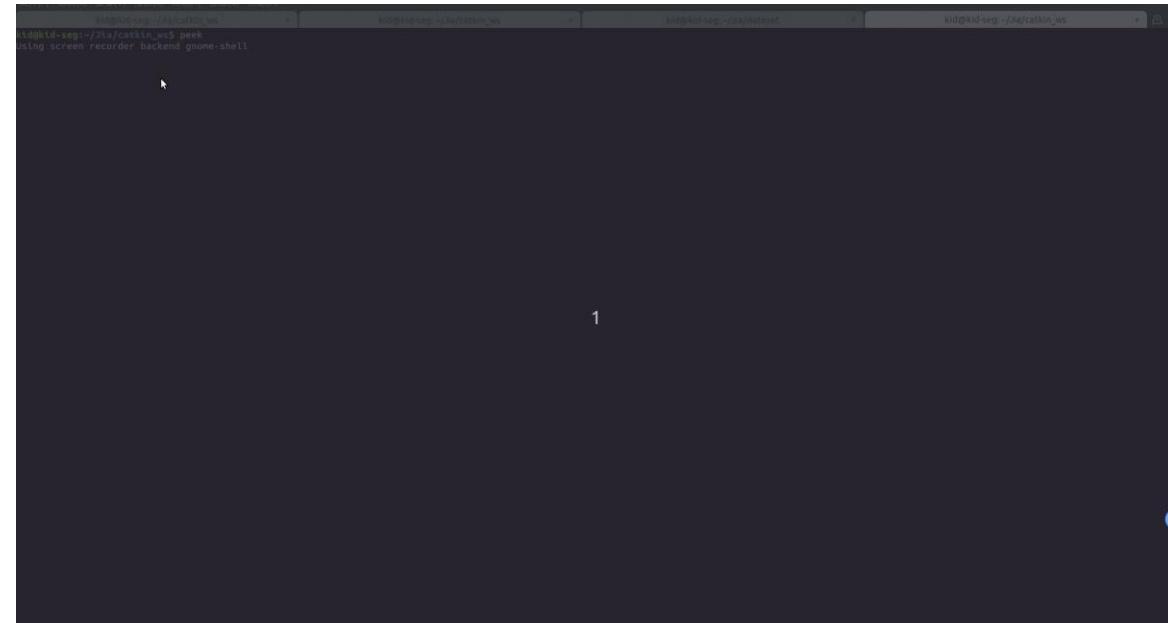
2. Build A-LOAM

Clone the repository and catkin_make:

```
cd ~/catkin_ws/src  
git clone https://github.com/HKUST-Aerial-Robotics/A-LOAM.git  
cd ../  
catkin_make  
source ~/catkin_ws/devel/setup.bash
```

- 使用rosbag
 - source devel/setup.bash每次打开新的terminal都要执行一次
 - roslaunch aloam_velodyne aloam_velodyne_HDL_32.launch脚本依据雷达而定
 - rosbag play xxx.bag

使用本地的rosbagfile运行A-LOAM效果 →

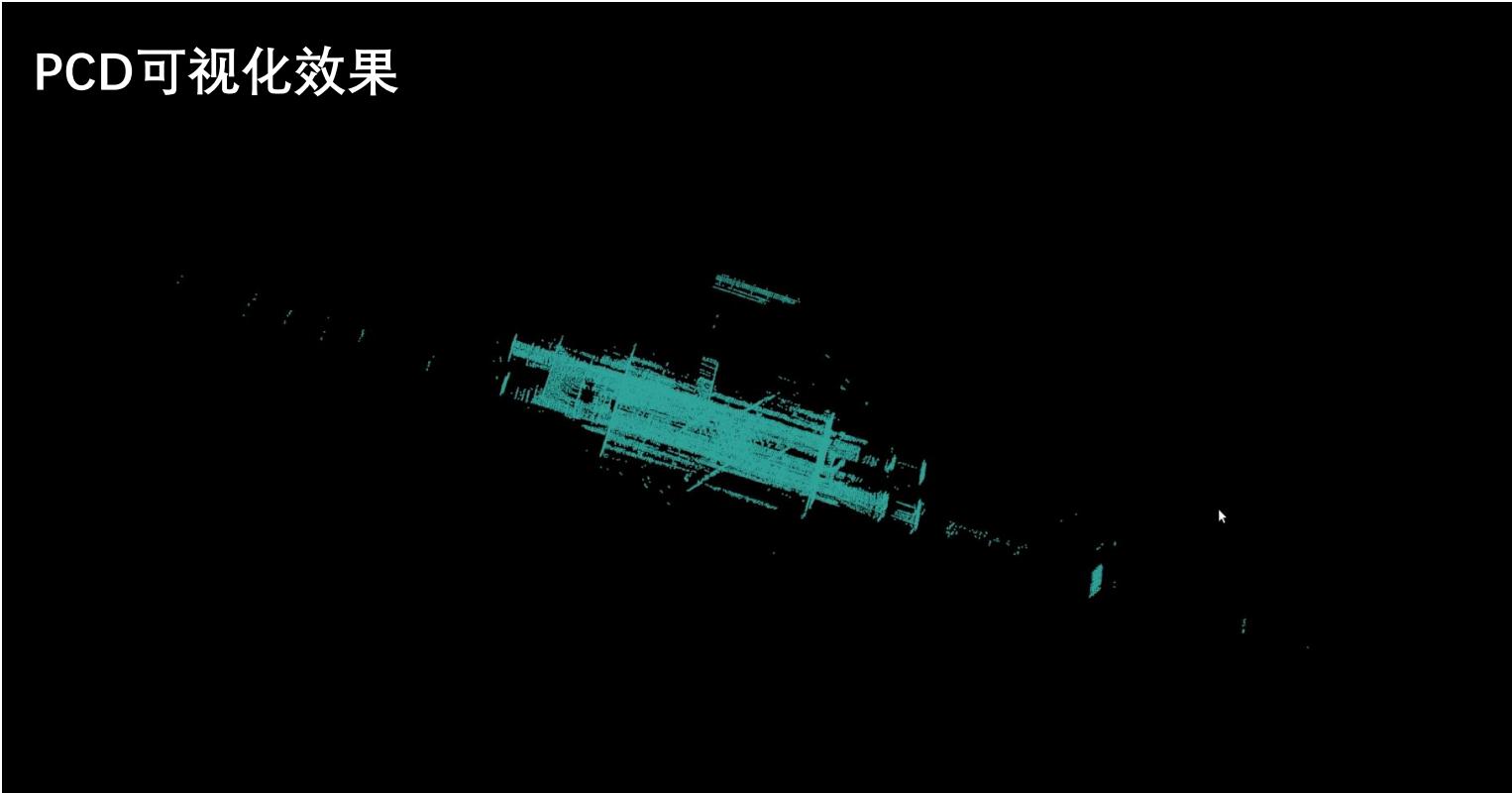


Part 1 | SLAM算法部署与评估

地图可视化

- 最后的点云也可以查看——pcd文件获取，CloudCompare或PCL_Visualizer查看PCD
 - A-LOAM：将之前获得的rosbagfile的点云数据解出来即可
 - rosrun pcl_ros bag_to_pcd xxx.bag /laser_cloud_map pcd
 - 运行上面代码前开一个terminal启动ros： roscore
 - 使用CloudCompare或pcl_viewer ***.pcd可以查看点云，最后一帧是叠起来的点云

PCD可视化效果



★ 实验：基于NAS中给出的bag文件，配置SLAM环境，并在本地回放bag，利用A-LOAM算法对每帧点云数据进行位姿估计。得到保存的每一帧pcd和odom文件，以及一个KITTI格式的pose.txt—用于EVO评估。.

» SLAM-EVO

EVO工具用来评估SLAM的位姿估计结果与真值（Ground Truth, GT）的误差。

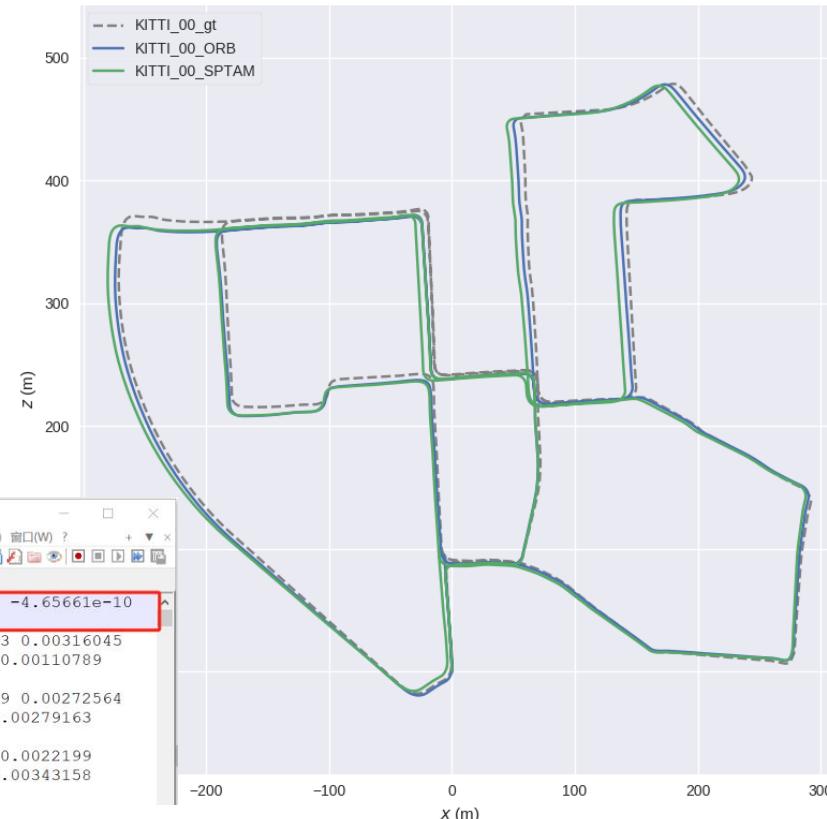
基准结果一般通过RTK-GPS，或高精度的测量设备得到，但成本高昂，且部署繁琐。

相当于车辆/传感器在移动过程中，通过GPS测量真值与Slam算法估计结果之间求差异。

KITTI 位姿数据格式（数据的存储形式）：



```
D:\data\SemanticKITTI\dataset\poses\00.txt - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
00.txt
1 1 9.31323e-10 -3.27418e-11 0 -9.31323e-10 1 -4.65661e-10
7.45058e-09 1.09139e-11 -9.31323e-10 1 0
2 0.999991 -0.00316351 -0.00274942 -0.00135393 0.00316045
0.999994 -0.00111659 -0.0248245 0.00275294 0.00110789
0.999996 0.672716
3 0.999969 -0.00274641 -0.00740973 -0.00497909 0.00272564
0.999992 -0.00281192 -0.0381271 0.0074174 0.00279163
0.999969 1.37469
4 0.999925 -0.00226129 -0.0120078 -0.0149304 0.0022199
0.999992 -0.00345849 -0.0500735 0.0120155 0.00343158
0.999922 2.08592
```



» SLAM-EVO

EVO主要包含两个度量：绝对位姿误差APE, absolute pose error

Corresponding poses are directly compared between **estimate** and **reference** given a pose relation. Then, statistics for the whole trajectory are calculated. This is useful to test the global consistency of a trajectory. 度量的是轨迹的全局一致性。

evo_ape tum reference.txt estimate.txt --align

APE

The absolute pose error is a metric for investigating the global consistency of a SLAM trajectory

APE is based on the absolute relative pose between two poses $P_{ref,i}, P_{est,i} \in \text{SE}(3)$ at timestamp i :

$$E_i = P_{est,i} \ominus P_{ref,i} = P_{ref,i}^{-1} P_{est,i} \in \text{SE}(3)$$

where \ominus is the inverse compositional operator, which takes two poses and gives the relative pose [Lu-1997]. You can use different pose relations to calculate the APE:

- `metrics.PoseRelation.translation_part`
 - this uses the translation part of E_i
 - $APE_i = \|\text{trans}(E_i)\|$
- `metrics.PoseRelation.rotation_angle_(rad/deg)`
 - uses the rotation angle of E_i
 - $APE_i = |\text{angle}(\log_{\text{SO}(3)}(\text{rot}(E_i)))|$
 - $\log_{\text{SO}(3)}(\cdot)$ is the inverse of $\exp_{\text{so}(3)}(\cdot)$ (Rodrigues' formula)
- `metrics.PoseRelation.rotation_part`
 - this uses the rotation part of E_i
 - $APE_i = \|\text{rot}(E_i) - I_{3 \times 3}\|_F$
 - unit-less
- `metrics.PoseRelation.full_transformation`
 - this uses the full relative pose E_i
 - $APE_i = \|E_i - I_{4 \times 4}\|_F$
 - unit-less

Then, different statistics can be calculated on the APEs of all timestamps, e.g. the RMSE:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N APE_i^2}$$

» SLAM-EVO

EVO主要包含两个度量：

相对位姿误差RPE,

Relative pose error

Relative pose error compares motions ("pose deltas"). This metric gives insights about the local accuracy, i.e. the drift. 度量的是轨迹的局部差异，也就是局部精度和漂移。

RPE

The relative pose error is a metric for investigating the local consistency of a SLAM trajectory

RPE compares the relative poses along the estimated and the reference trajectory. This is based on the delta pose difference:

$$E_{i,j} = \delta_{est,i,j} \ominus \delta_{ref,i,j} = (P_{ref,i}^{-1} P_{ref,j})^{-1} (P_{est,i}^{-1} P_{est,j}) \in \text{SE}(3)$$

You can use different pose relations to calculate the RPE from timestamp i to j :

- `metrics.PoseRelation.translation_part`
 - this uses the translation part of $E_{i,j}$
 - $RPE_{i,j} = \|\text{trans}(E_{i,j})\|$
- `metrics.PoseRelation.rotation_angle_(rad/deg)`
 - uses the absolute angular error of $E_{i,j}$
 - $RPE_{i,j} = |(\text{angle}(\log_{SO(3)}(\text{rot}(E_{i,j}))))|$
 - $\log_{SO(3)}(\cdot)$ is the inverse of $\exp_{SO(3)}(\cdot)$ (Rodrigues' formula)
- `metrics.PoseRelation.rotation_part`
 - this uses the rotation part of $E_{i,j}$
 - $RPE_{i,j} = \|\text{rot}(E_{i,j}) - I_{3\times 3}\|_F$
 - unit-less
- `metrics.PoseRelation.full_transformation`
 - this uses the full delta pose difference $E_{i,j}$
 - $RPE_{i,j} = \|E_{i,j} - I_{4\times 4}\|_F$
 - unit-less

Then, different statistics can be calculated on the RPEs of all timestamps, e.g. the RMSE:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{\forall i,j} RPE_{i,j}^2}$$

Part 5 | 激光SLAM

★旋转和平移的量纲不一样，是否能够进行统一度量？

EVO评估

» SLAM-EVO

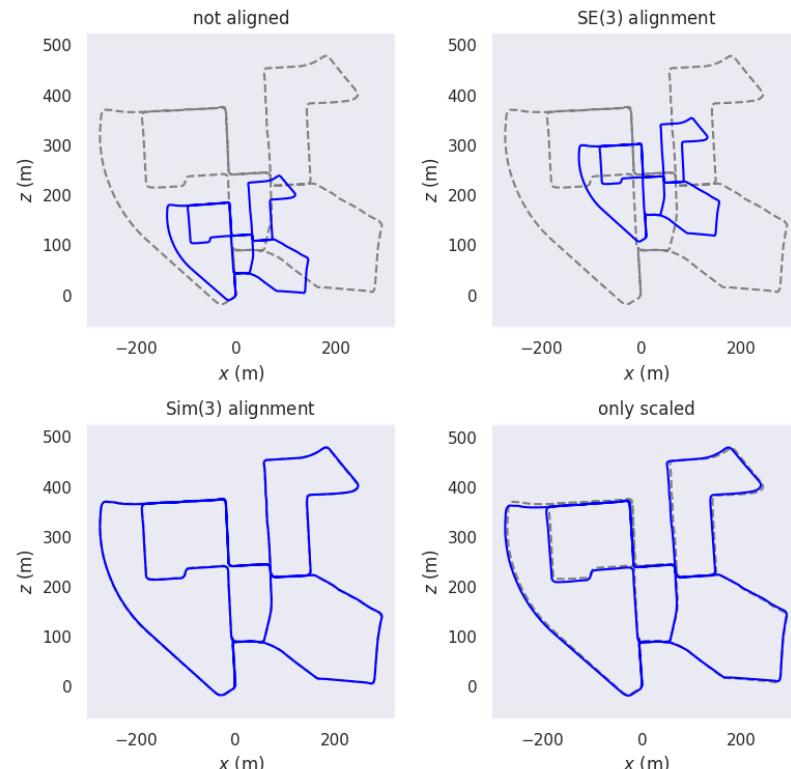
	计算目标	单位
A P E	Full transformation	unit-less
	traslation part	m
	rotation part	unit-less
	rotation angle in radians	rad
	rotation angle in degrees	deg
	point distance	m

```
class StatisticsType(Enum):
    rmse = "rmse" # root mean squared error
    mean = "mean"
    median = "median"
    std = "std"
    min = "min"   ★简单的统计指标能够反馈位姿误差?
    max = "max"
    sse = "sse" # sum of squared errors
```

统计指标

	计算目标	单位	delta_unit
R P E	Full transformation	unit-less	frames
	traslation part	m	
	rotation part	unit-less	
	rotation angle in radians	rad	
	rotation angle in degrees	deg	

其它操作：
同步
对齐
放缩



- SLAM算法保存rosbag数据
 - 在执行SLAM算法前执行rosbag play前执行rosbag record即可
 - A-LOAM: rosbag record -o bag_out /laser_cloud_map /aft_mapped_to_init
 - 得到存有里程计信息的rosbagfile，并将其放置到evo文件夹下
- 在conda虚拟环境里安装evo (github.com) 与jupyter (pip install jupyter)
- 启动conda环境，执行jupyter notebook开启jupyter服务，在弹出的网页上调试附件代码

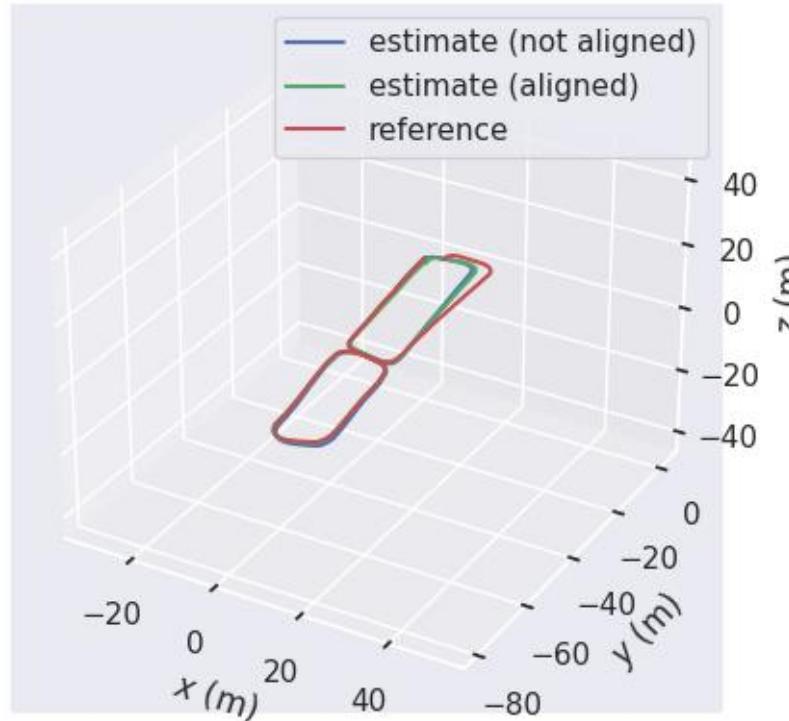


Jupyter服务打开的文件夹

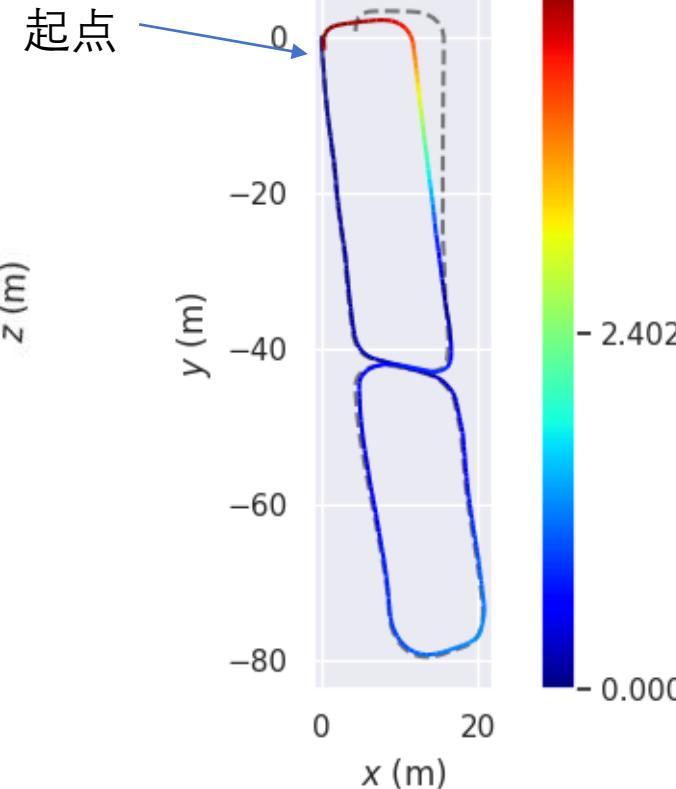
Part 5 | SLAM算法部署与评估

APE估计中，轨迹末端呈现累积误差不断增大。

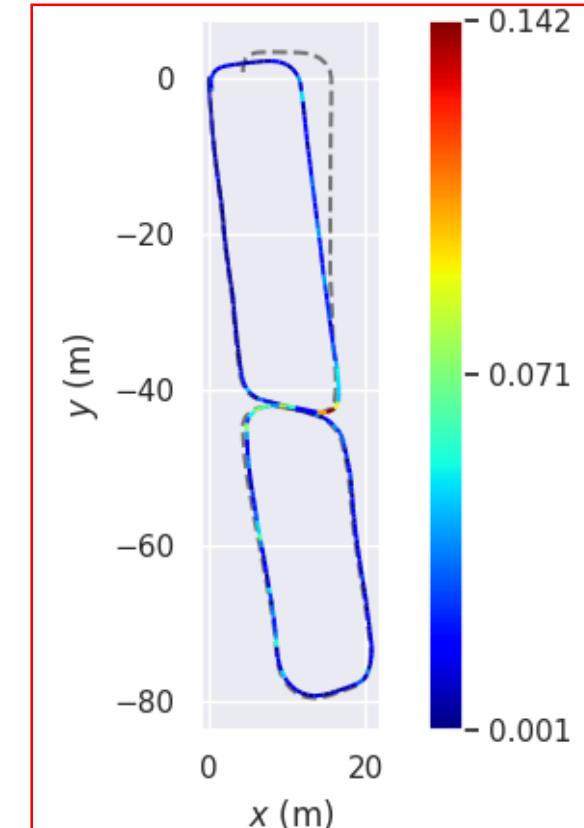
EVO评估



EVO可视化的POSE示意图



APE对比SLAM



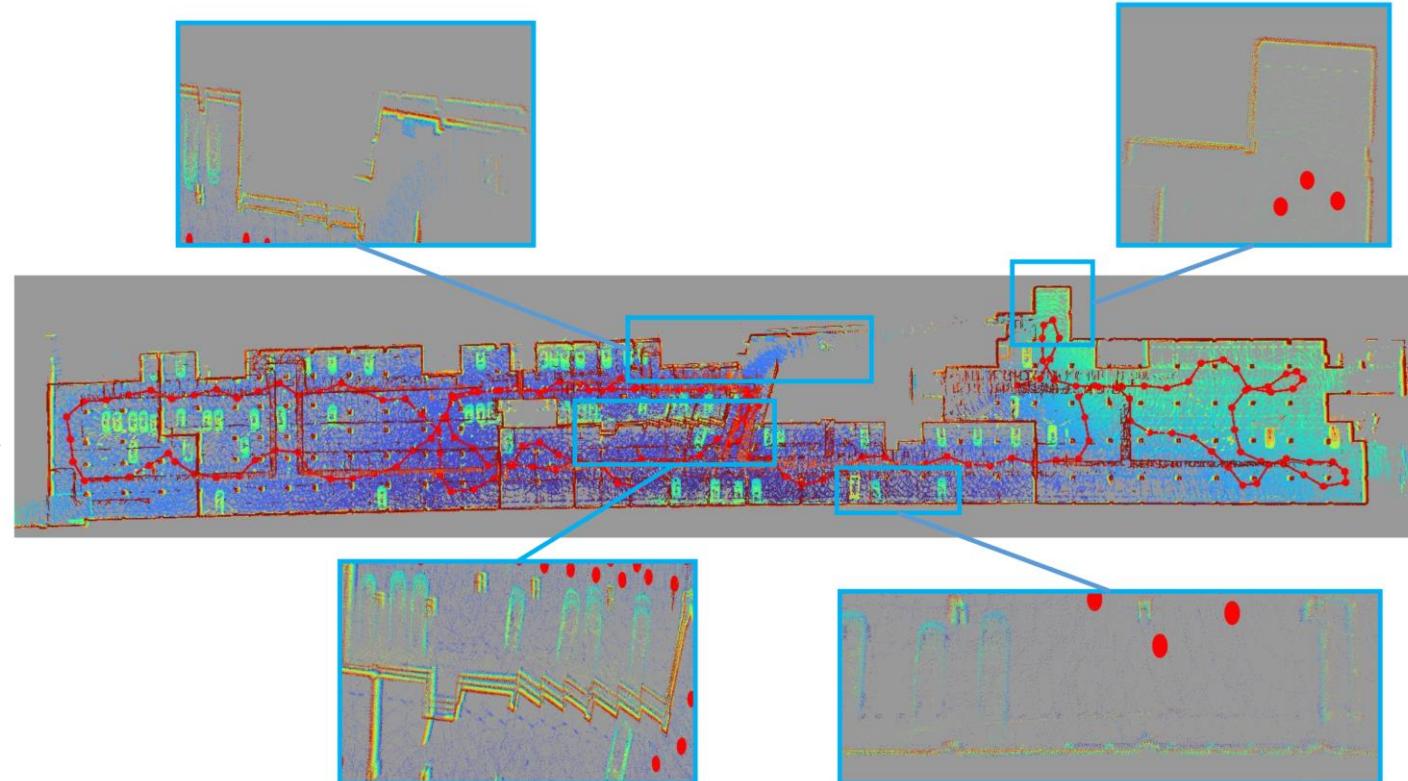
PRE对比SLAM

结果较差的pose位置，应该用来引导交互slam调整

★实验：基于KITTI格式的pose.txt，与NAS给出的真值，在EVO环境进行轨迹可视化对比，以及APE和RPE的估计。

› 大规模一致性映射

- 即使最先进的SLAM框架，在进行大规模环境的一致性映射时也会面临挑战。这通常需要根据传感器和环境特性仔细调整超参数，而对于非专家用户来说非常困难。



› 局部和全局不一致

- 在没有良好超参数选择的情况下，自动SLAM系统可能会产生局部和全局地图不一致的问题，例如墙壁和地板的重叠加倍和弯曲。

› 人工参与的SLAM的局限性

- 虽然已有研究允许用户通过GUI交互式地添加约束来校正2D地图，但这些方法在3D映射中的效率较低，且只能添加新约束而不能细化现有约束。

Part 5 | 交互SLAM--Interactive SLAM

交互优化

» ROS需求环境一览

各版本要求

Ubuntu 18 + ros1 melodic

官方ros安装教程

<http://wiki.ros.org/Installation/Ubuntu?distro=melodic>

中文安装教程：

<https://www.ncynl.com/archives/201906/3147.html>

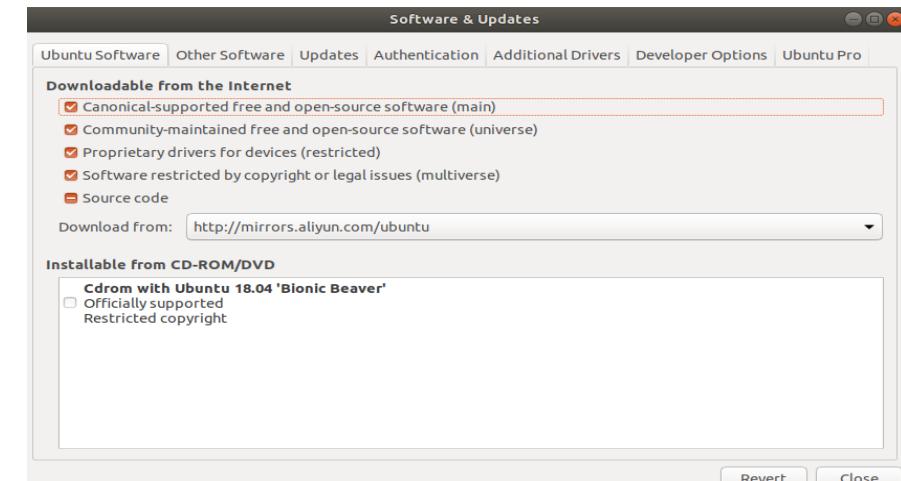
一些版本要求（重要！）：

cmake 3.14以上

gcc 8.0及以上

» Ubuntu 18系统

配置软件更新和源



允许“stricted,” “universe,” and “multiverse.”，可以选择阿里云，然后点击Close，然后Reload。

Part 5 | 交互SLAM--Interactive SLAM

交互优化

安装中国的源

```
sudo sh -c './etc/lsb-release && echo "deb http://mirrors.ustc.edu.cn/ros/ubuntu/ $DISTRIB_CODENAME main" > /etc/apt/sources.list.d/ros-latest.list'
```

设置key

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key  
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

更新

```
sudo apt-get update
```

安装Desktop-Full和rosinstall便利工具

```
sudo apt-get install ros-melodic-desktop-full
```

```
sudo apt install python-rosdep python-rosinstall python-rosinstall-
```

解决依赖

```
sudo rosdep init  
rosdep update
```

环境设置

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc  
source ~/.bashrc
```

测试运行

```
roscore
```

```
ty@ty-ubuntu18:~$ roscore  
... logging to /home/ty/.ros/log/bb9003b8-9102-11ef-9306-94bb434fc4ac/roslaunch-t  
y-ubuntu18-29106.log  
Checking log directory for disk usage. This may take a while.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
  
started roslaunch server http://ty-ubuntu18:39405/  
ros_comm version 1.14.13  
  
SUMMARY  
=====
```

ctrl + c 可退出

```
PARAMETERS  
* /rosdistro: melodic  
* /rosversion: 1.14.13  
  
NODES  
  
auto-starting new master  
process[master]: started with pid [29131]  
ROS_MASTER_URI=http://ty-ubuntu18:11311/  
  
setting /run_id to bb9003b8-9102-11ef-9306-94bb434fc4ac  
process[rosout-1]: started with pid [29143]  
started core service [/rosout]
```

Part 5 | 交互SLAM--Interactive SLAM

交互优化

1

版本不同，很有可能会出现报错！！！

注意一些依赖的版本(重要):

cmake 3.14以上

gcc 8.0及以上

glog-0.6.0

GTSAM 4.0.3

ceres-solver2.2.0

需支持c++17，含absl库

需支持GLSL3.3

如确实出现相关报错，先查看附录

cuda安装

for ROS noetic(针对ros noetic版本的配置)

sudo apt-get install libglm-dev libglfw3-dev

sudo apt-get install libsuitesparse-dev libeigen3-dev

sudo apt-get install ros-noetic-geodesy ros-noetic-pcl-ros
ros-noetic-nmea-msgs

Install g2o from original source code

git clone https://github.com/RainerKuemmerle/g2o.git

cd g2o

mkdir build

cd build

cmake ..

make

sudo make install

Part 5 | 交互SLAM--Interactive SLAM

交互优化

如有报错，可以参考附录，或自行搜索

```
cd ~/catkin_ws/src  
git clone https://github.com/koide3/ndt_omp  
# on melodic  
# git clone https://github.com/koide3/ndt_omp -b melodic  
git clone https://github.com/koide3/hdl_graph_slam  
git clone https://github.com/koide3/odometry_saver  
git clone https://github.com/SMRT-AIST/fast_gicp --  
recursive  
git clone https://github.com/SMRT-AIST/interactive_slam --  
recursive  
  
cd ~/catkin_ws  
catkin_make -DCMAKE_BUILD_TYPE=Release
```

2

```
cd ~/catkin_ws/src  
git clone https://github.com/koide3/ndt_omp  
# on melodic  
# git clone https://github.com/koide3/ndt_omp -b melodic  
git clone https://github.com/koide3/hdl_graph_slam  
git clone https://github.com/koide3/odometry_saver  
git clone https://github.com/SMRT-AIST/fast_gicp --recursive  
git clone https://github.com/SMRT-AIST/interactive_slam --  
recursive
```

3

```
cd ~/catkin_ws  
catkin_make -DCMAKE_BUILD_TYPE=Release  
# 若采用catkin_make_isolated，则别忘了执行以下命令  
source ~/catkin_ws/devel_isolated/setup.bash
```

Part 5 | 交互SLAM--Interactive SLAM

交互优化

- 可以在交互slam可读的转储格式下，使用自定义算法(ALOAM)来保存建图结果。运行`run.launch`后，建图结果保存在目录/tmp/dump下。
- 例子中，使用从rosbag文件中使用ALOAM生成的里程数据建图。例子的rosbag文件：R132.bag(实验提供的数据包)
- 1. 安装

```
cd catkin_ws/src  
git clone https://github.com/RobustFieldAutonomyLab/LeGO-  
LOAM.git  
git clone https://github.com/koide3/odometry_saver.git
```

需要编辑LeGO-LOAM中的“utility.h”，适配实际的传感器（默认数据可跳过）。

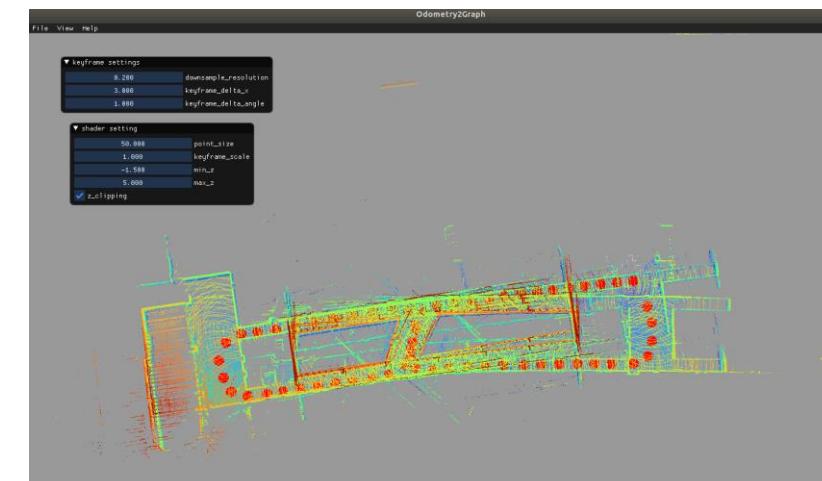
- 2. 生成里程数据

```
roslaunch odometry_saver online.launch dst_directory:=/home/kid/Jia/aloam  
points_topic:=/velodyne_points odom_topic:=/aft_mapped_to_init endpoint_frame:=aft_mapped  
origin_frame:=camera_init # 根据实际算法调整  
roslaunch aloam run.launch  
rosbag play --clock hdl_400.bag  
rosrun interactive_slam odometry2graph # 回放完毕，导入到odometry2graph会得到一张图
```

保存里程数据：Menubar -> File -> Open -> ROS -> Choose /tmp/odometry

- 3. 保存图数据
 - 菜单-> File -> Save -> 选择导出路径(导出的数据会在下一步使用)

```
1 graph_directory  
2   |- 000000  
3   |   |- cloud.pcd  
4   |   |- data  
5   |- 000001  
6   |   |- cloud.pcd  
7   |   |- data  
8   |- 000002  
9   ...  
10  |- graph.g2o  
11  |- graph.g2o.kernels
```



Part 5 | 交互SLAM--Interactive SLAM

第五代化

- 1、启动交互slam

rosrun interactive_slam interactive_slam

- 2、打开图目录

菜单 -> File -> Open -> New map -> 选择上一步导出的目录

- 3、自动闭环检测(如果有明显断路, 先进行人工闭环, 见后一页)

Menubar菜单 -> Graph -> Automatic loop detection

配置如下参数: (保持点击并拖动可以进行数值调整)

Scan matching

Method: NDT_OMP

Fitness score thresh: 0.5

Loop detection

Distance thresh: 20 # 限制了关键点形成闭环的距离, 搜索半径增大or限制对key frame搜索的最小间距

Accum distance thresh: 25 or 0.5 (dense loop closing) # 限制相邻闭环间距, 防止过于密集

Robust kernel

Kernel type:Cauchy

点解start。完成后, 再设置Accum distance thresh = 0.5, 进行密集闭环。

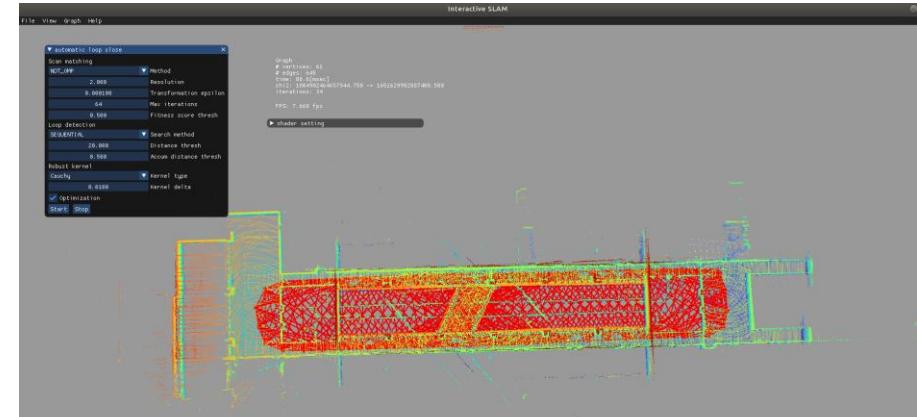
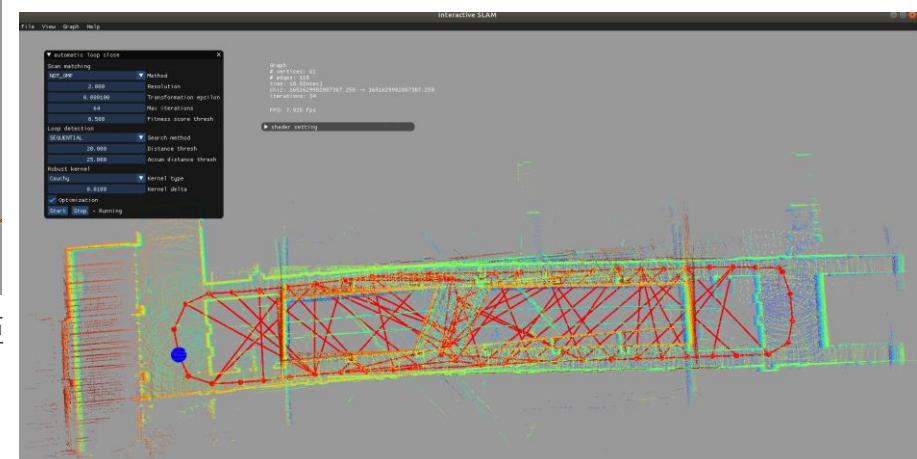
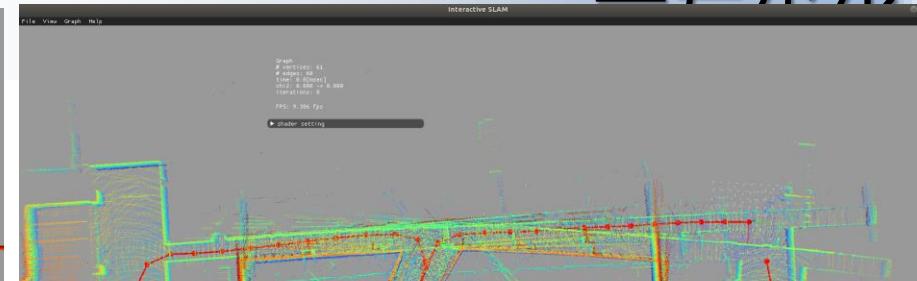
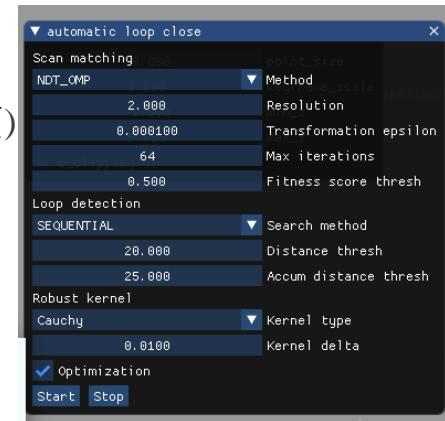
- 4、保存优化后的图和点云

Menubar菜单 -> File -> Save -> Save map data -> Choose the destination directory

Menubar菜单 -> File -> Save -> Export point cloud -> Choose the destination filename

可以通过兼容PCD的查看器来打开, 比如pcl_viewer和CloudCompare。

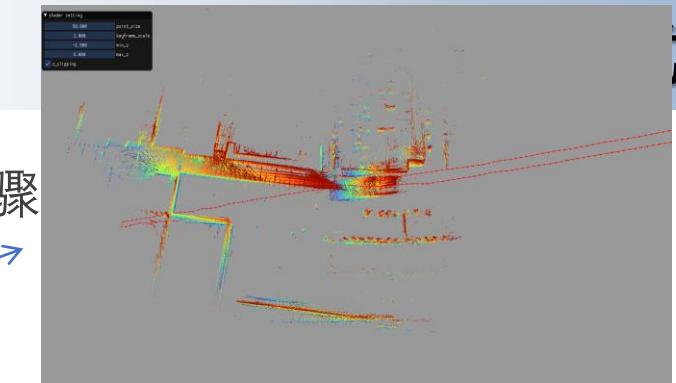
<https://blog.csdn.net/QLeelq/article/details/123063347> 安装pcl_viewer



Part 5 | 交互SLAM--Interactive SLAM

- 后面的例子使用的官方数据，优化实验数据可以有选择地借用后面的步骤
- 下载数据，打开交互slam并加载“centrair_101_graph”

Menubar -> File -> Open -> New map -> Choose "centrair_101_graph"

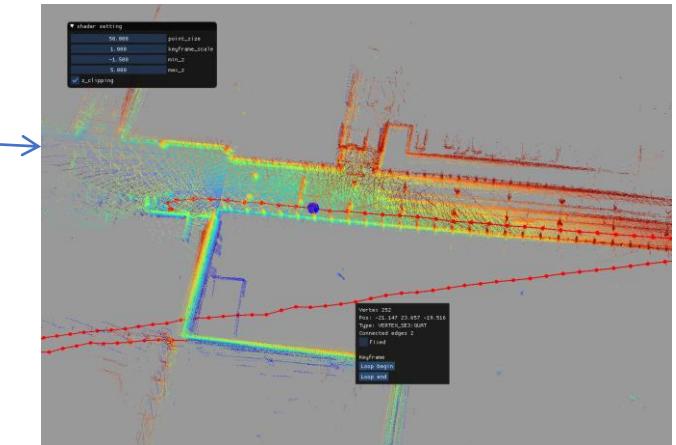


人工闭环

选择两个关键帧(一个点代表)来限制进行闭环。

右键一个关键帧 -> Loop begin -> Right click another one -> Loop end

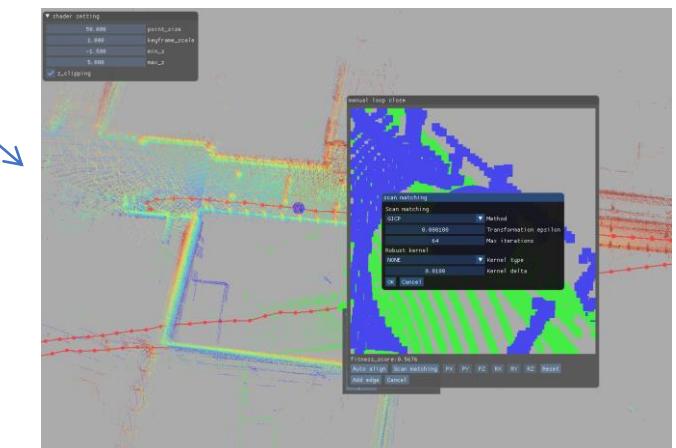
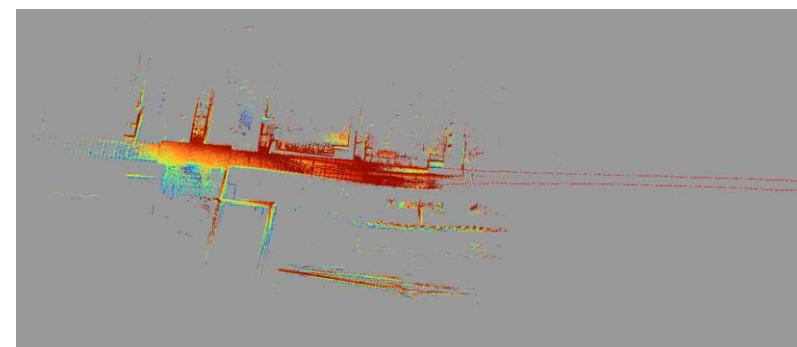
在弹出的界面中，观察效果，猜测纹理较多则效果好。点击Auto align，
点击ok，使用基于FPFH的位姿估计作初步闭环预测。



粗略配准关键帧后，进行精细registration，点击“Scan matching”，并ok。

检查配准是否精细。点击“Add edge”增加相对位姿边到相对位姿图。

优化之后，可以看到地图的全局连续性大幅提升。

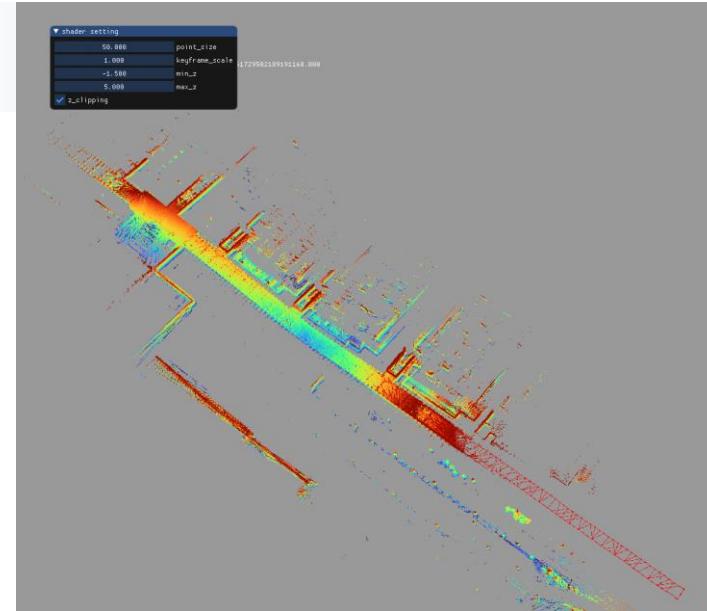


Part 5 | 交互SLAM--Interactive SLAM

优化

- 闭环检测(理解)

- 交互SLAM实现了手动和自动闭环检测过程，以修正里程计漂移。用户可以通过图形用户界面(GUI)选择闭环候选点，系统通过快速点特征直方图(FPFH)估算初始变换，并使用GICP(广义迭代最近点)算法进行精细配准。在初步手动调整后，自动闭环检测根据空间位置和匹配得分自动寻找可能的闭环。这个迭代的闭环修正过程显著提高了地图的全局一致性。

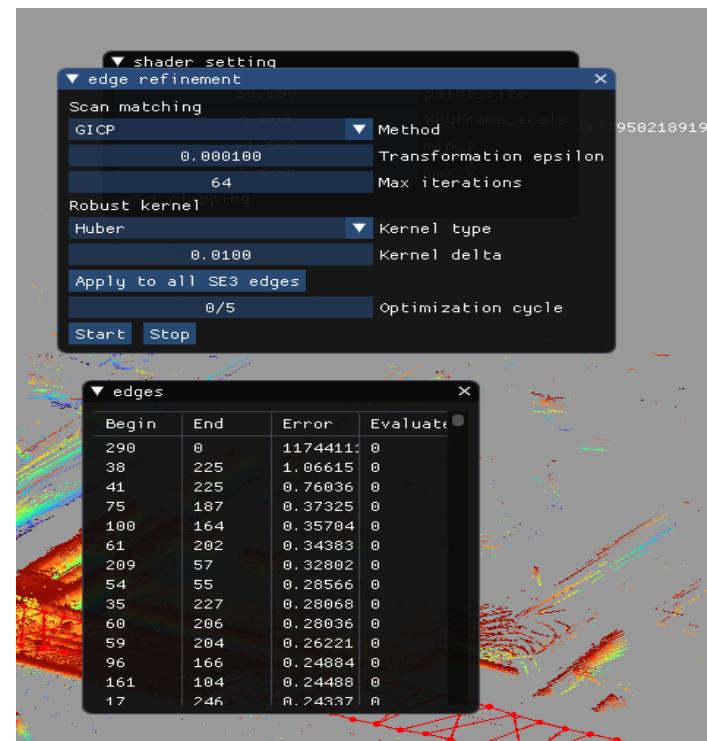


- 自动闭环

- 提升局部连续性(两层墙等)，可以进行自动闭环操作。
菜单 -> Graph -> Automatic loop detection -> Start

- 边优化

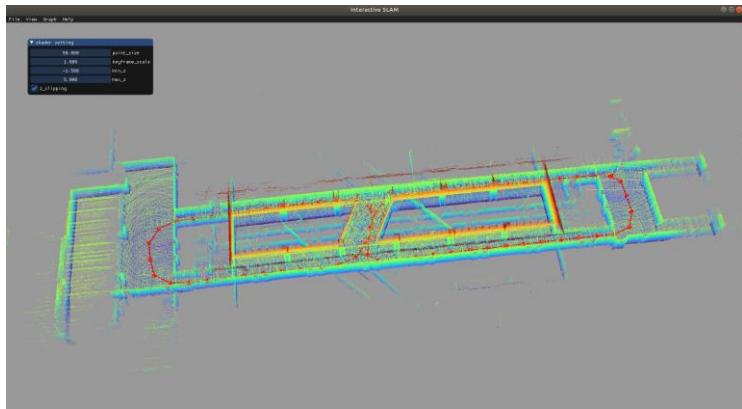
- 由于里程数据的一些错误，仍可能存在大的畸变。
调整相对位姿约束可以优化地图质量。
- 菜单 -> Graph -> Edge refinement -> Start
- (问题：Optimization cycle只能调整第二个数字，第一个调整不了)
- 这个过程选择一个误差较大的边，
并在与该边连接的关键帧之间执行扫描匹配。



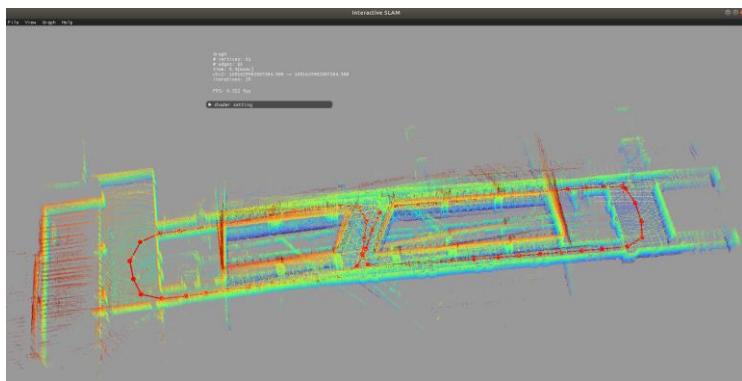
Part 5 | 交互SLAM--Interactive SLAM

交互优化

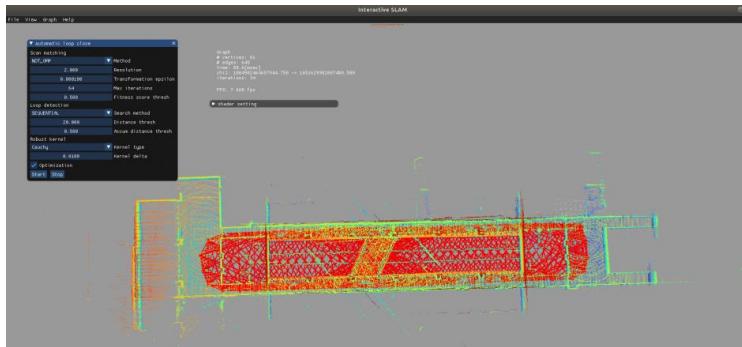
GT



Poor(LeGOLOAM)



Good
(Interactive_SLAM Refine)



Hausdorff Distance 衡量两个点集之间的“最坏情况”距离。(一般越小越好)

Hausdorff Distance 通过找到每个点集到另一个点集的最近距离，并取这些距离的最大值来衡量点集之间的距离。它反映了两个点集之间的最远“误差”，即最远的一对点之间的距离。

Chamfer Distance 通过计算两个点集之间所有点到另一集合的平均距离来衡量它们的相似度。(一般越小越好)

Chamfer Distance 将每个点到另一点集的最近距离进行平均，而不是取最大值。这种度量方式相对较为平滑，不容易受到少数离群点的影响，更关注整体的距离分布情况。

Hausdorff Distance (GT vs Poor): 31.88846875223777

Chamfer Distance (GT vs Poor): 0.7011567544004397

Hausdorff Distance (GT vs Good): 31.97795257866959

Chamfer Distance (GT vs Good): 0.5176966735946409

- 在运行命令`sudo apt-get install ros-noetic-geodesy ros-noetic-pcl-ros ros-noetic-nmea-msgs`时报错`unable to locate.....`

原因：命令是对于noetic版本的ros，需要修改版本

输入命令`echo \$ROS_DISTRO`查看ros版本，输出版本为melodic，则

```
sudo apt-get install ros-melodic-geodesy ros-melodic-pcl-ros ros-melodic-nmea-msgs
```

- 搭建g2o时报错，需要cmake3.14以上

可以选择安装3.17，适合cuda10.2，当然更高版本应该也可以。

更改版本的方法参考：<https://huaweicloud.csdn.net/635641d9d3efff3090b5c780.html>

- 报错`fatal error: filesystem: No such file or directory #include <filesystem>`

原因：gcc版本低于8.0，可以选择安装8.0或者9.0，以下提供8.0安装参考

```
sudo apt-get install gcc-8 g++-8
```

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-8 90 --slave /usr/bin/g++ g++ /usr/bin/g++-8 --slave /usr/bin/gcov gcov /usr/bin/gcov-8
```

```
gcc --version
```

其中90是优先级数，越高越优先，覆盖掉以前的版本。

注意！！如果你已经cmake过了，除了更换gcc版本，还需要把之前编译的文档删除，不然会出现新的报错。删除方法：

```
cd /path/to/your/build
```

```
rm -rf *
```

```
cmake ..
```

```
make
```

- 报错找不到absl包

Could not find a package configuration file provided by "absl" with the following names:

abslConfig.cmake

absl-config.cmake

解决方法，安装absl库：

<https://blog.csdn.net/k1419197516/article/details/130118003>

- ceres-solver的报错

- git命令如果连接失败，则尝试网页下载ceres-solver2.2.0，再发送到linux系统下
- make的时候报错，针对这个报错的原因是：

libgoogle-glog版本的问题，

查阅博客：

https://blog.csdn.net/qq_60243596/article/details/128501339

```
Plain Text 默认主题
1 /usr/include/glog/logging.h:638:9: error: ambiguous overload for 'operator<<' (operand types are '
2 (*os) << v;
3 ~~~~~^~~
4 In file included from /usr/include/c++/8/istream:39,
5                 from /usr/include/c++/8/sstream:38, 这个过程选择一个误差较大的边，并在与该边连接的
6 .....
7 usr/include/c++/8/ostream:577:5: note: candidate: 'std::basic_ostream<char, _Traits>& std::operator<<(basic_ostream<char, _Traits>& __out, const unsigned char* __s)
8         operator<<(basic_ostream<char, _Traits>& __out, const unsigned char* __s)
9         ^~~~~~
10 这个过程选择一个误差较大的边，并在与该边连接的关键帧之间执行扫描匹配。internal/ceres/CMakeFiles/ceres
11 make[2]: *** [internal/ceres/CMakeFiles/ceres_internal.dir/cuda_sparse_matrix.cc.o] Error 1
12 CMakeFiles/Makefile2:2186: recipe for target 'internal/ceres/CMakeFiles/ceres_internal.dir/all' failed
13 make[1]: *** [internal/ceres/CMakeFiles/ceres_internal.dir/all] Error 2
14 Makefile:157: recipe for target 'all' failed
15 make: *** [all] Error 2
```

```
ty@ty-ubuntu18:~/catkin_ws/src/ceres-solver/build$ sudo apt-get install libgogl
gle-glog-dev
[sudo] password for ty:
Reading package lists... Done
Building dependency tree
Reading state information... Done
libgoogle-glog-dev is already the newest version (0.3.5-1).
The following packages were automatically installed and are no longer required
:
  fonts-liberation2 fonts-opensymbol gir1.2-goa-1.0
  gir1.2-gst-plugins-base-1.0 gir1.2-gstcamer-1.0 gir1.2-gudev-1.0
  gir1.2-snapd-1 gir1.2-udisks-2.0 grilo-plugins-0.3-base gstreamer1.0-gtk3
  libcdr-0.1-1 libclucene-contribs1v5 libclucene-core1v5 libcmis-0.5-5v5
  libdazzle-1.0-0 libe-book-0.1-1 libedataserver1.2-2 libeot0
  libepubgen-0.1-1 libetonyek-0.1-1 libevent-2.1-6 libexiv2-14
  libfreerdp-client-2 libfreerdp2-2 libgc1c2 libgee-0.8-2 libgexiv2-2
  libgom-1.0-0 libgpmp6 libgpod-common libgpod4 liblangtag-common
  liblangtag1 liblirc-client0 liblua5.3-0 libmediaart-2.0-0 libmxmlsec1-0.1-1
  libodfgen-0.1-1 libqqwing2v5 librevenge-0.0-0 libsgutils2-2 libssh-4
  libvncclient1 libwinpr2-2 libxapian30 libxmlsec1 libxmlsec1-nss lp-solve
  media-player-info python3-mako python3-markupsafe syslinux syslinux-common
  syslinux-legacy usb-creator-common
```

- 在执行`catkin_make -DCMAKE_BUILD_TYPE=Release`的时候报错

- 如果最后的报错类似于：Error 2 Invoking "make -j32 -l32" failed，则往前查阅日志信息，找到最初的报错输出，再定位问题。

- 解决办法：让代码支持c++17，在cmake文件中增加代码。

```
## 先找到cmake文件
```

```
ls /src/CMakeLists.txt -l # 查看发现是软连接，改对应的源文件
```

```
sudo vim /opt/ros/melodic/share/catkin/cmake/toplevel.cmake
```

在代码文件中添加以下，使用c++17

```
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
```

```
1 /usr/local/include/g2o/core/matrix_structure.h:57:19: error: 'std::string_view' has not been declared
2   57 |     bool write(std::string_view filename) const;
3   |     ^~~~~~
4 In file included from /usr/local/include/g2o/core/sparse_block_matrix.h:42,
5           from /usr/local/include/g2o/core/marginal_covariance_cholesky.h:34,
6           from /usr/local/include/g2o/core/linear_solver.h:33,
7           from /usr/local/include/g2o/core/block_solver.h:35,
8           from /home/ty/catkin_ws/src/hdl_graph_slam/src/hdl_graph_slam/graph_slam.cpp:9:
9 /usr/local/include/g2o/core/matrix_structure.h:57:19: error: 'std::string_view' has not been declared
10  57 |     bool write(std::string_view filename) const;
11  |     ^~~~~~
12 In file included from /usr/local/include/g2o/core/base_fixed_sized_edge.h:39,
13           from /usr/local/include/g2o/core/base_binary_edge.h:30,
14           from /usr/local/include/g2o/types/slam3d/types_slam3d.h:31,
15           from /home/ty/catkin_ws/src/hdl_graph_slam/src/hdl_graph_slam/graph_slam.cpp:16:
16 /usr/local/include/g2o/stuff/tuple_tools.h: In function 'void g2o::tuple_apply_i(F&&, T&, int)':
17 /usr/local/include/g2o/stuff/tuple_tools.h:41:46: error: 'tuple_size_v' is not a member of 'std';
```

- 或者在运行命令的时候增加参数指定(推荐方法)

```
catkin_make_isolated -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_STANDARD=17
```

Part 5 | 附录-4

- 启动交互slam报错

需要支持GLSL3.3,

<https://blog.csdn.net/ssm1122/article/details/139656718>

解决办法:

export MESA_GL_VERSION_OVERRIDE=3.3

- lego_loam报错

原因: GTSAM版本与Eigen冲突。

解决办法:

CMAKELISTS.TXT增加一句代码,
且更换GRSAM为4.0.3版本。重新编译。

参考:

https://blog.csdn.net/weixin_44023934/article/details/113539840

- 最后编译成功后, 别忘设置环境变量

source ~/catkin_ws/devel_isolated/setup.bash

- 还有其他报错请尝试自行查阅资料解决

```

9 error : failed to link program
10 error: linking with uncompiled/unspecialized shadererror: linking with uncompiled/unspecialized sh
11 construct solver: lm_var cholmod
12 done
13 error : failed to compile shader /home/ty/catkin_ws/src/interactive_slam/data/shader/rainbow.vert
14 0:1(10): error: GLSL 3.30 is not supported. Supported versions are: 1.10, 1.20, 1.30, 1.40, 1.00 E
15
16 error : failed to compile shader /home/ty/catkin_ws/src/interactive_slam/data/shader/rainbow.frag
17 0:1(10): error: GLSL 3.30 is not supported. Supported versions are: 1.10, 1.20, 1.30, 1.40, 1.00 E
18
19 error : failed to link program
20 error: linking with uncompiled/unspecialized shadererror: linking with uncompiled/unspecialized sh

```

```

1 error: static assertion failed: Error: GTSAM was built against a different version of Eigen
2     GTSAM_EIGEN_VERSION_WORLD==EIGEN_WORLD_VERSION &&
3
4 In file included from /usr/include/pcl-1.8/pcl/sample_consensus/sac_model.h:52,
5                 from /usr/include/pcl-1.8/pcl/sample_consensus/sac.h:45,
6                 from /usr/include/pcl-1.8/pcl/sample_consensus/ransac.h:44,
7                 from /usr/include/pcl-1.8/pcl/registration/icp.h:45,
8                 from /home/ty/catkin_ws/src/LeGO-LOAM/LeGO-LOAM/include/utility.h:24,
9                 from /home/ty/catkin_ws/src/LeGO-LOAM/LeGO-LOAM/src/transformFusion.cpp:33:
10 /usr/include/pcl-1.8/pcl/sample_consensus/model_types.h: In function 'void __static_initialization
11

```

开始实验

<https://github.com/hahakid/3D-point-cloud-processing-and-visualization-practice/tree/main/lecture3>



北京航空航天大学
BEIHANG UNIVERSITY