



# SLAM应用介绍及经典算法 (LOAM) 解析

王晓晨

2022年04月02日



# 目录

**1. 为什么要做SLAM?**

**2. 现有SLAM算法**

**3. LOAM算法解析**

# 为什么要做SLAM?

有前（钱）景？

难以落地！

SLAM 算法工程师 [北京]

30-60K·14薪 经验不限 | 学历不限 刘女士 | HRBP

宽凳科技

互联网 | B轮 | 100-499人



餐补, 节日福利, 零食下午茶, 股票期权, 定期体检, 带...

SLAM Engineer [上海 闵行区·吴泾]

30-40K·13薪 1-3年 | 硕士 秦女士 | HRG

非夕机器人

计算机软件 | B轮 | 100-499人



节日福利, 定期体检, 五险一金, 员工旅游, 年终奖, 零...

SLAM算法工程师 [北京 海淀区·西北旺]

50-80K·15薪 经验不限 | 硕士 吴先生 | 副总裁

地平线

互联网 | C轮 | 1000-9999人



节日福利, 通讯补贴, 五险一金, 定期体检, 股票期权, ...

视觉SLAM算法工程师 [上海 嘉定区·黄渡]

21-40K·14薪 1-3年 | 硕士 宗先生 | 招聘者

舵敏智能

汽车零部件 | 天使轮 | 0-20人



年终奖, 定期体检, 五险一金, 带薪年假, 股票期权, 节...

SLAM算法工程师 [北京]

30-60K·14薪 经验不限 | 硕士 秦女士 | HR

北京鉴智机器人

计算机软件 | A轮 | 100-499人



股票期权, 补充医疗保险, 节日福利, 餐补, 五险一金, ...

SLAM算法工程师 [苏州 昆山市·玉山]

20-40K 5-10年 | 硕士 张先生 | 人力资源总监

三一重工

企业服务 | 已上市 | 10000人以上



节日福利, 年终奖, 五险一金, 带薪年假, 专利奖金, 免...

✓ L0: 无自动化

✓ L1: 辅助驾驶

✓ L2: 部分自动化

□ L3: 有条件自动化

□ L4: 高度自动化

□ L5: 全自动化

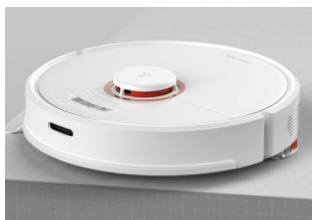


# 为什么要做SLAM?

## ●应用领域



服务机器人



自动驾驶



三维重建





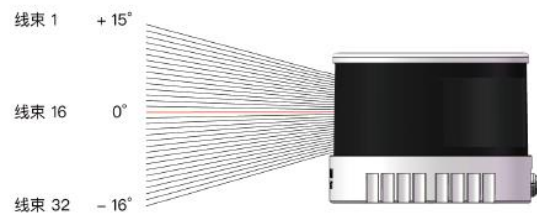
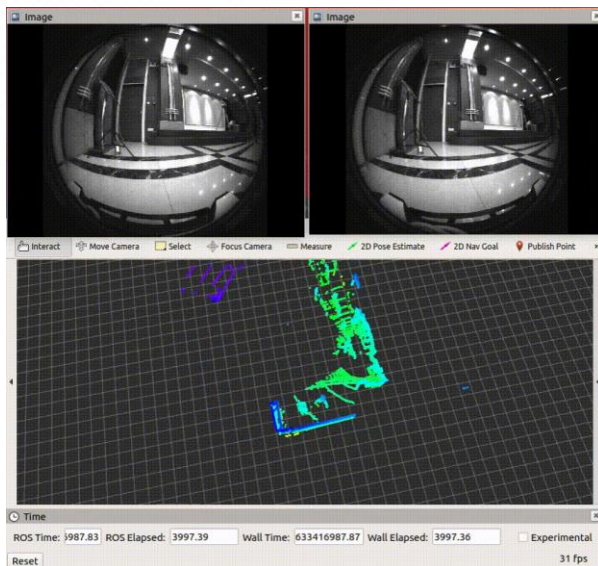
# 为什么要做SLAM?

## ● 背景

各种移动载体对准确、稳定的定位需求日益增加，但现有的**GNSS + IMU**定位方式在室内环境、停车场以及结构复杂的高架桥等**信号易丢失的场景**中难以满足实际生产需要。

## ● 主要思路

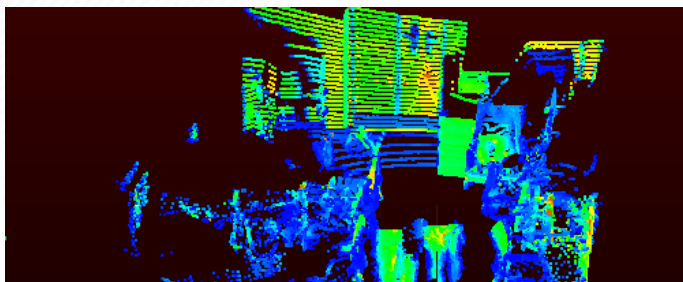
引入**自主定位源**，根据传感器的类型主要分为**视觉**（单目/双目相机、深度相机）**激光**（2D/3D激光雷达）两类。



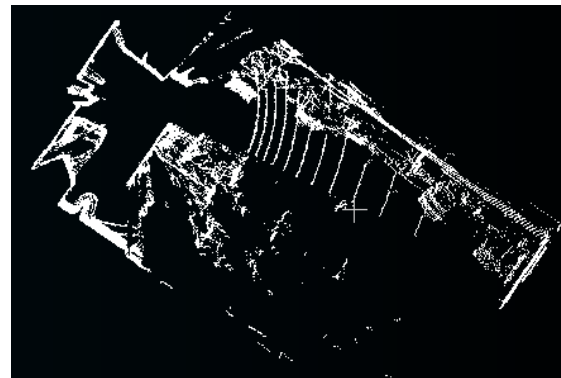
# 为什么要做SLAM?

## ● 存在问题

- ❑ 复杂场景中**动态物体**会引起噪声干扰，导致出现错误的匹配，定位信息与构建的环境地图出现严重偏差。



- ❑ 单一传感器具有很大的局限性，如**视觉**的定位导航方案容易受到**光照**明暗变化、**天气**条件、**视角**等因素影响，**激光雷达**则对这些条件更加鲁棒，但也存在**纹理**信息不足的问题。



# 为什么要做SLAM?

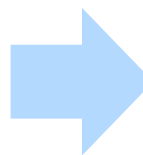
## ● 解决方案

多源数据融合



充分利用相机、激光雷达、惯导等传感器数据的优势，在保证**实时**的前提下，搭建一个**高精度、鲁棒**的多源数据定位导航解决方案。

利用深度学习技术解决现有定位导航方案中存在的难点



提前引入动态目标的**语义信息**作为先验知识，更加稳定、准确地剔除**动态粗差点**的干扰，提高定位建图精度。

# 目录

1. 为什么要做SLAM?

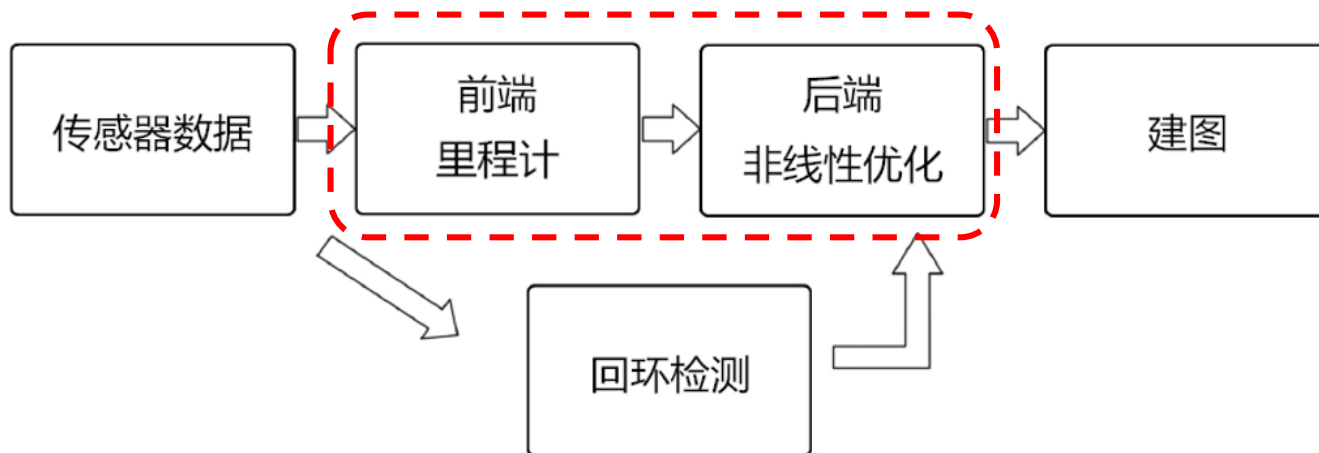
2. 现有SLAM算法

3. LOAM算法解析



# 现有SLAM算法

## ● 模块组成



## ● 前端里程计

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

<https://blog-ss>

## ● 后端非线性优化



















$$\min_x = \frac{1}{2} \|f(x)\|_2^2$$

梯度下降法

高斯牛顿法

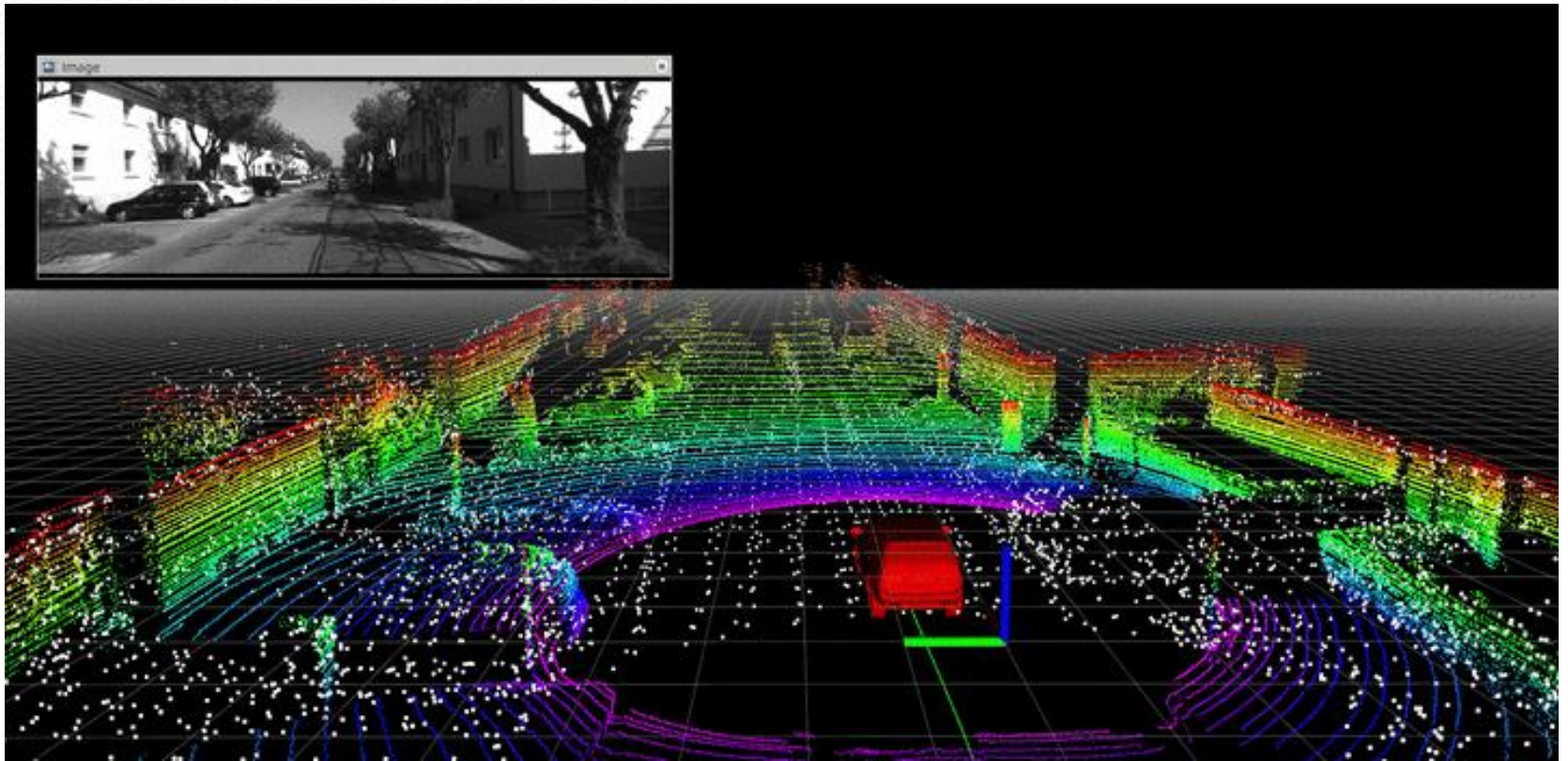
列文伯格-马夸特 (LM)

# 现有SLAM算法

	Method	Setting	Code	Translation	Rotation	Runtime	Environment	Compare
1	<a href="#">SOFT2</a>			0.53 %	0.0009 [deg/m]	0.1 s	4 cores @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
I. Cvišić, I. Marković and I. Petrović: <a href="#">Recalibrating the KITTI Dataset Camera Setup for Improved Odometry Accuracy</a> . European Conference on Mobile Robots (ECMR) 2021.								
2	<a href="#">V-LOAM</a>			0.54 %	0.0013 [deg/m]	0.1 s	2 cores @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
J. Zhang and S. Singh: <a href="#">Visual-lidar Odometry and Mapping: Low drift, Robust, and Fast</a> . IEEE International Conference on Robotics and Automation(ICRA) 2015.								
3	<a href="#">LOAM</a>			0.55 %	0.0013 [deg/m]	0.1 s	2 cores @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
J. Zhang and S. Singh: <a href="#">LOAM: Lidar Odometry and Mapping in Real- time</a> . Robotics: Science and Systems Conference (RSS) 2014.								
4	<a href="#">TVL-SLAM+</a>	 		0.56 %	0.0015 [deg/m]	0.3 s	1 core @ 3.0 Ghz (C/C++)	<input type="checkbox"/>
C. Chou and C. Chou: <a href="#">Efficient and Accurate Tightly-Coupled Visual-Lidar SLAM</a> . IEEE Transactions on Intelligent Transportation Systems 2021.								
5	<a href="#">GLIM</a>			0.59 %	0.0015 [deg/m]	0.1 s	GPU @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
K. Koide, M. Yokozuka, S. Oishi and A. Banno: <a href="#">Globally Consistent 3D LiDAR Mapping with GPU-accelerated GICP Matching Cost Factors</a> . IEEE Robotics and Automation Letters 2021.								
6	<a href="#">CT-ICP</a>		<a href="#">code</a>	0.59 %	0.0014 [deg/m]	0.06 s	1 core @ 3.5 Ghz (C/C++)	<input type="checkbox"/>
P. Dellenbach, J. Deschaud, B. Jacquet and F. Goulette: <a href="#">CT-ICP: Real-time Elastic LiDAR Odometry with Loop Closure</a> . arXiv e-prints 2021.								
7	<a href="#">SDV-LOAM</a>			0.60 %	0.0015 [deg/m]	0.05 s	4 cores @ 3.0 Ghz (C/C++)	<input type="checkbox"/>
8	<a href="#">HELO</a>			0.61 %	0.0018 [deg/m]	0.1 s	8 cores @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
9	<a href="#">CT-ICP-test</a>			0.62 %	0.0015 [deg/m]	0.06 s	1 core @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
10	<a href="#">wPICP</a>			0.62 %	0.0015 [deg/m]	0.1 s	1 core @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
11	<a href="#">HMLQ</a>			0.62 %	0.0014 [deg/m]	0.2s	1 core @ 2.5 Ghz (Matlab)	<input type="checkbox"/>
12	<a href="#">HMLQ-whu</a>			0.63 %	0.0014 [deg/m]	0.2 s	1 core @ 2.5 Ghz (Matlab)	<input type="checkbox"/>
13	<a href="#">filter-reg</a>			0.65 %	0.0016 [deg/m]	0.1 s	1 core @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
14	<a href="#">SOFT-SLAM</a>			0.65 %	0.0014 [deg/m]	0.1 s	2 cores @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
I. Cvišić, J. Cesić, I. Marković and I. Petrović: <a href="#">SOFT-SLAM: Computationally Efficient Stereo Visual SLAM for Autonomous UAVs</a> . Journal of Field Robotics 2017.								
15	<a href="#">MULLS</a>		<a href="#">code</a>	0.65 %	0.0019 [deg/m]	0.08 s	4 cores @ 2.2 Ghz (C/C++)	<input type="checkbox"/>
Y. Pan, P. Xiao, Y. He, Z. Shao and Z. Li: <a href="#">MULLS: Versatile LiDAR SLAM via Multi- metric Linear Least Square</a> . IEEE International Conference on Robotics and Automation (ICRA) 2021. .								
16	<a href="#">LO-TEST</a>			0.66 %	0.0021 [deg/m]	0.1 s	1 core @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
17	<a href="#">SMTD-LO</a>		<a href="#">code</a>	0.66 %	0.0020 [deg/m]	0.3 s	1 core @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
18	<a href="#">PICP</a>			0.67 %	0.0018 [deg/m]	0.05 s	1 core @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
19	<a href="#">ELO</a>			0.68 %	0.0021 [deg/m]	0.005 s	GPU @ 2.6 Ghz (C/C++)(0.027s Jetson AGX)	<input type="checkbox"/>
X. Zheng and J. Zhu: <a href="#">Efficient LiDAR Odometry for Autonomous Driving</a> . IEEE Robotics and Automation Letters(RA- L) 2021.								
20	<a href="#">GLO</a>			0.69 %	0.0018 [deg/m]	0.1 s	1 core @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
21	<a href="#">IMLS-SLAM</a>			0.69 %	0.0018 [deg/m]	1.25 s	1 core @ >3.5 Ghz (C/C++)	<input type="checkbox"/>

# 现有SLAM算法

## ● LOAM



<https://github.com/HKUST-Aerial-Robotics/A-LOAM>



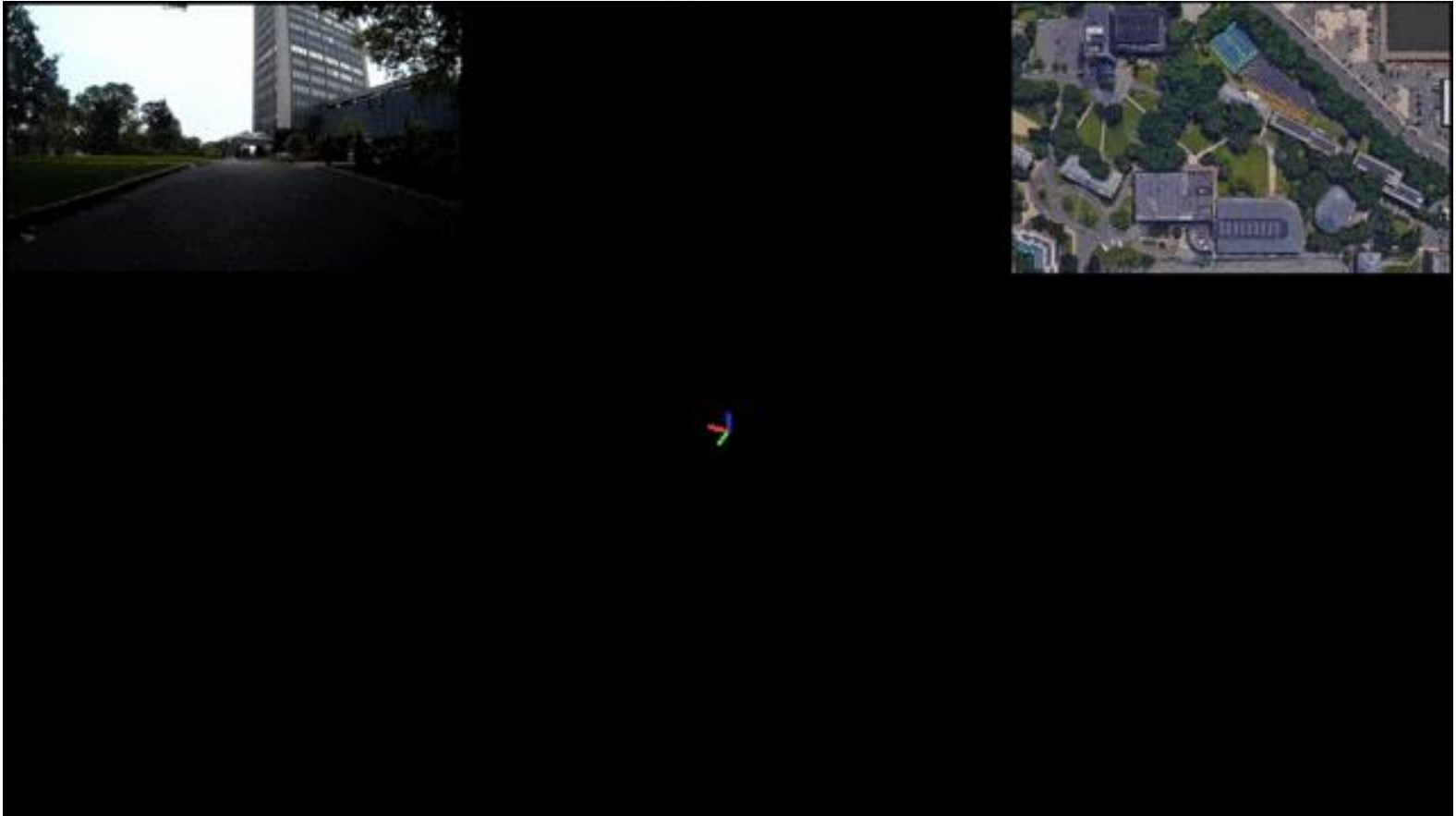
# 现有SLAM算法

## ● LOAM



# 现有SLAM算法

## ● LeGo-LOAM

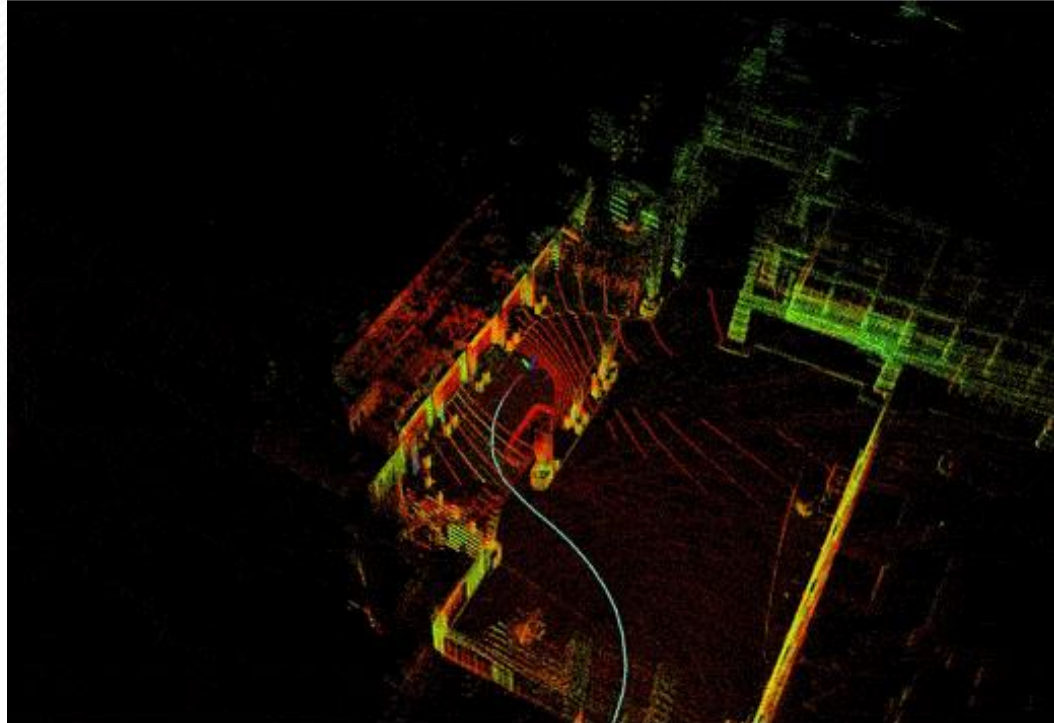


<https://github.com/RobustFieldAutonomyLab/LeGO-LOAM>



# 现有SLAM算法

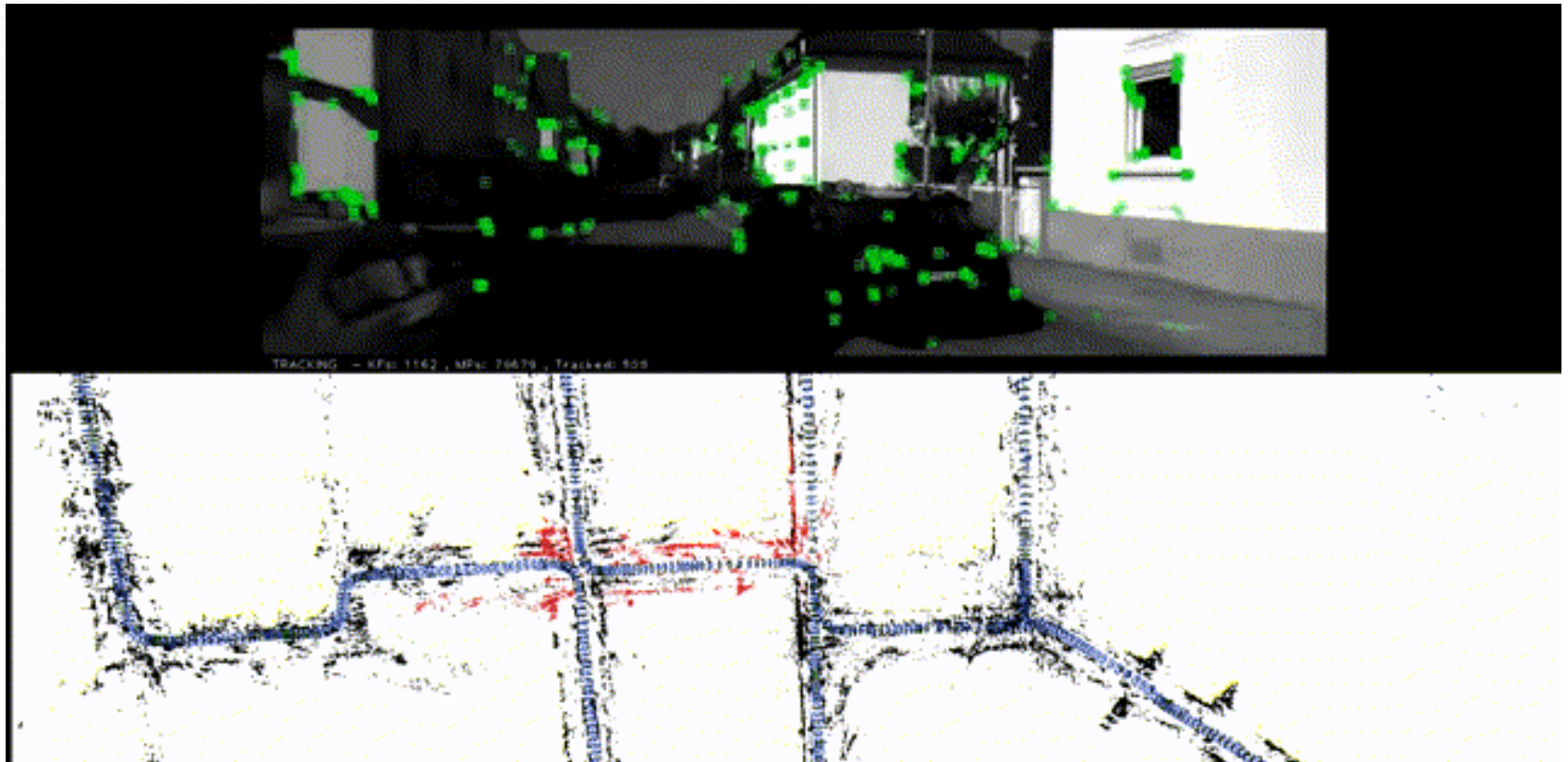
## ● LIO-SAM



<https://github.com/TixiaoShan/LIO-SAM>

# 现有SLAM算法

## ● ORB-SLAM-Monocular

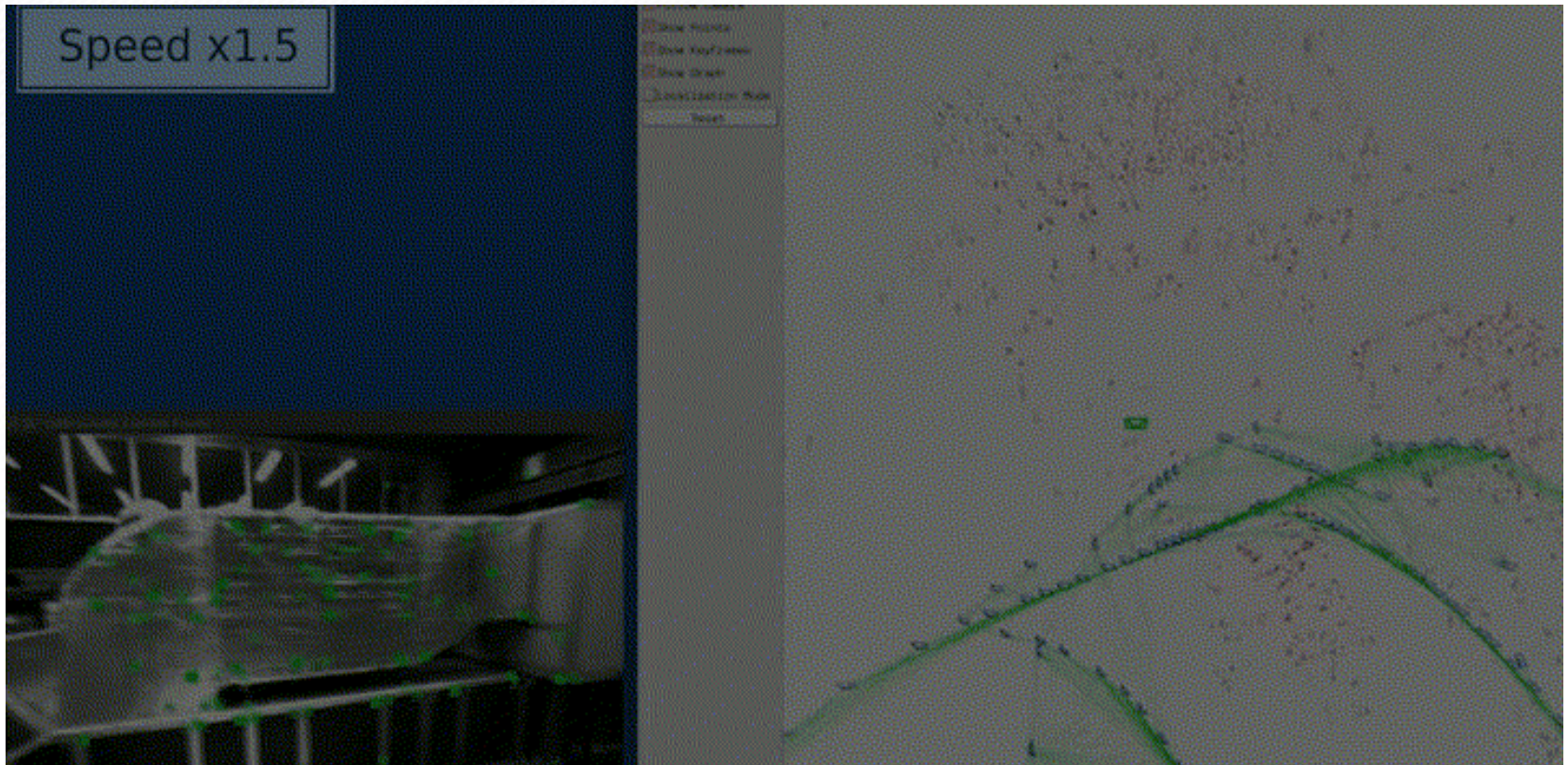


[https://github.com/raulmur/ORB\\_SLAM](https://github.com/raulmur/ORB_SLAM)



# 现有SLAM算法

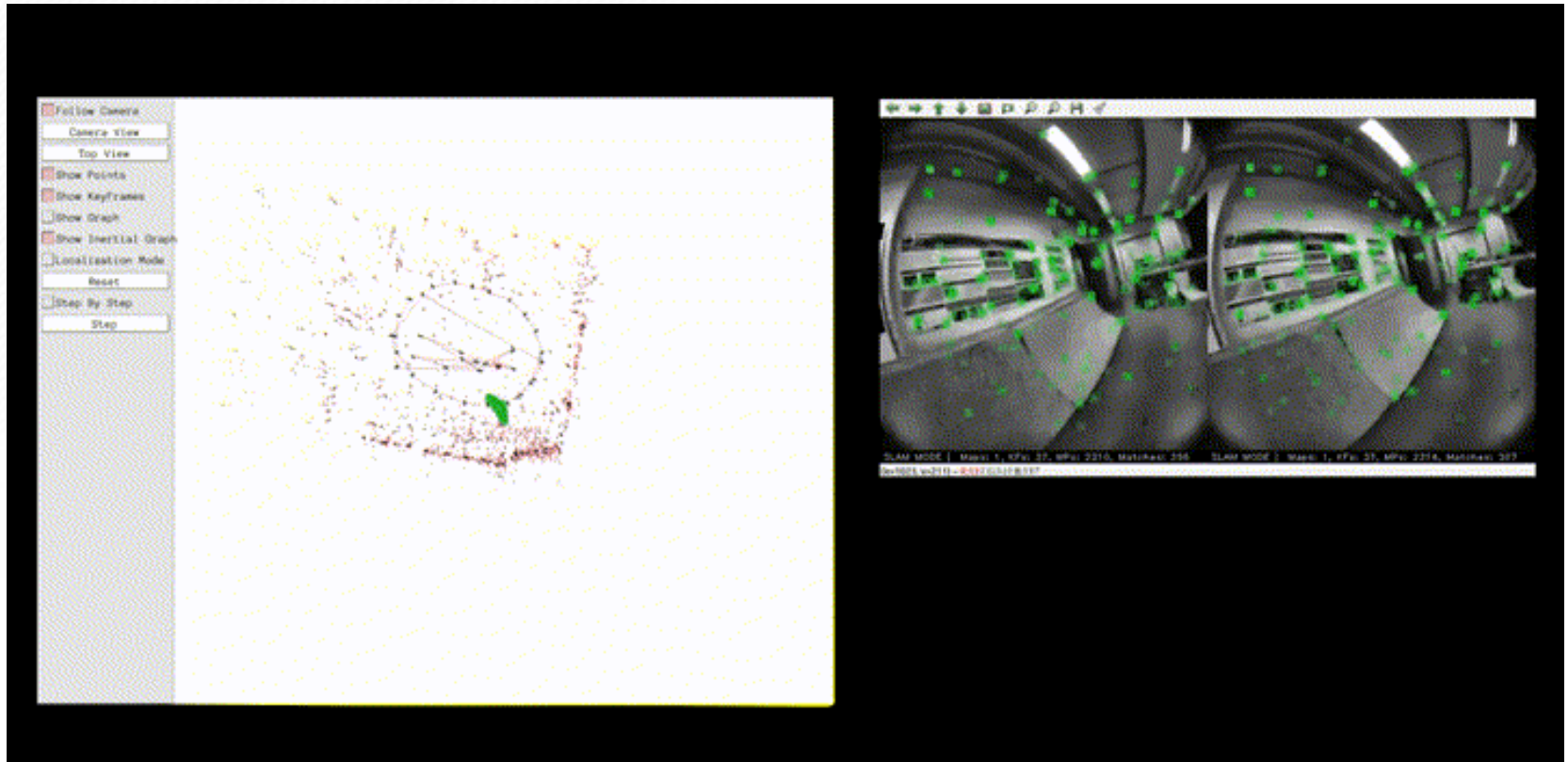
## ● ORB-SLAM2-Monocular, Stereo and RGB-D



[https://github.com/raulmur/ORB\\_SLAM2](https://github.com/raulmur/ORB_SLAM2)

# 现有SLAM算法

## ● ORB-SLAM3-Monocular, Stereo and RGB-D (pin-hole and fisheye lens)



# 目录

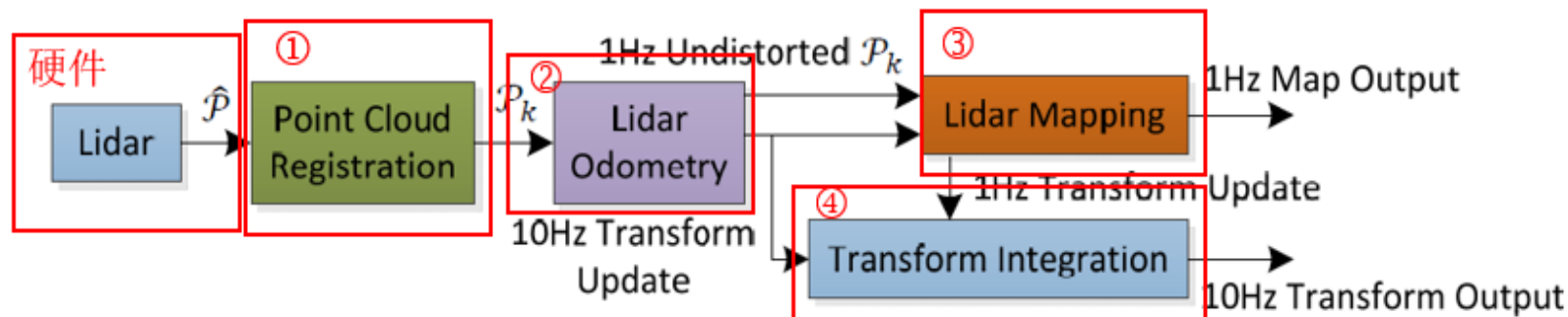
1. 为什么要做SLAM?

2. 现有SLAM算法

3. LOAM算法解析



# LOAM算法解析



- Lidar: 硬件采集数据, 不断发布点云数据 (10 HZ)。
- Point Cloud Registration: 接收原始点云数据, 进行预处理 (去除无效点)。接着, 根据点云平滑度提取特征点。
- Lidar Odometry: 高频、低精度的里程计来估计LiDAR的位姿, 并以计算结果去除扫描过程中产生的运动畸变。
- Lidar Mapping: 低频、高精度的里程计对上一模块产生的LiDAR位姿进行优化, 并转换到全局坐标系下。
- Transform Integration: 将Lidar Odometry和Lidar Mapping中得到的姿态信息放入到rviz中, 方便观看和处理。

# LOAM算法解析

## ► Point Cloud Registration

*Step1:* 对原始点云数据进行预处理，去除无效点。

传感器获得的数据可能在一些点的坐标中为无效值（没有值，可能是因为目标距离传感器太近或太远，或者因为反射导致传感器没有接收到回波脉冲），从而导致点的坐标为NaN。

*Step2:* 计算每个点的所属的scanID及在当前帧数据中的相对时间。

计算接收到点云数据在竖直方向上的角度，根据传感器的扫描线数及竖直方向上的角度间隔计算每个点所属的扫描线号。此外，计算出扫描点在当前帧数据中的相对时间，并对应存储。

*Step3:* 平滑度计算。

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

S为当前点前后各5个点组成的点集， $X_{(k,i)}^L$ 为当前点， $X_{(k,j)}^L$ 为点集S中的其他点。

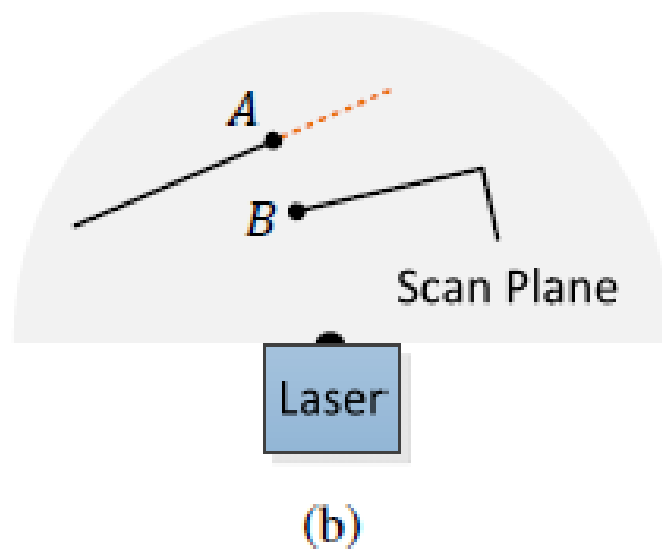
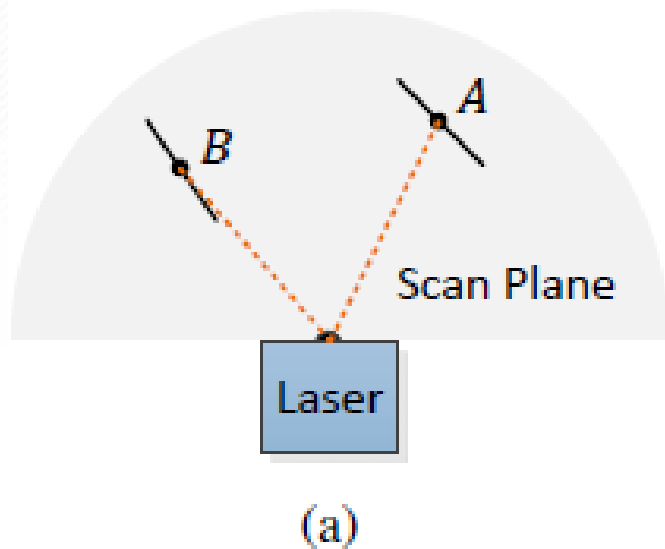
# LOAM算法解析

## Point Cloud Registration

Step4: 提取特征点。

为了让特征点能够均匀分布，文章中把每个扫描分为4个部分（代码中分为了6个部分），在每个子集区域的每条扫描线选取2个sharp点和4个flat点。

此外，还需要考虑如下两种情况：



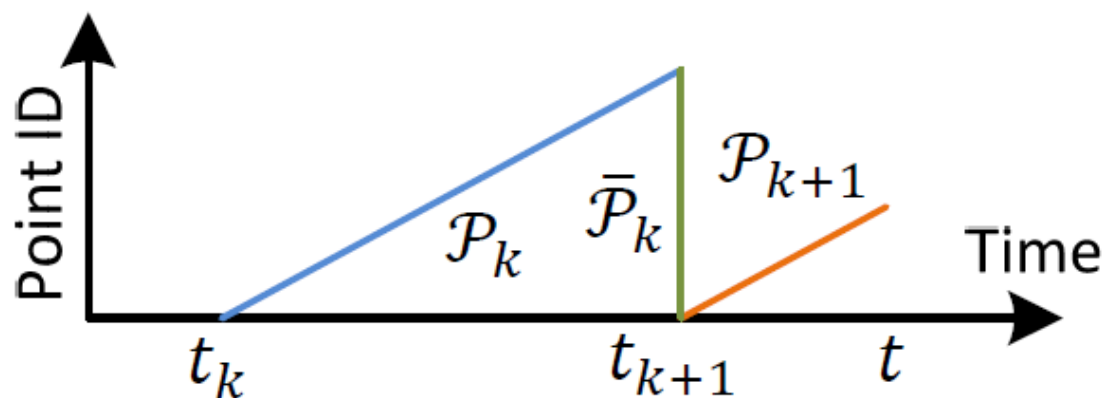
- 1) 避免选取位于与激光束平行的平面上的点，如左图(a)中的B点。
- 2) 避免选取被其他物体遮挡区域边缘的点，如右图(b)中的A点。

# LOAM算法解析

## ► Lidar Odometry

Step1: 寻找对应的特征点

如下图所示，假设一个扫描 $k$ 的开始时间戳为 $t_k$ ，在第 $k$ 帧扫描结束 $t_{k+1}$ 接收到的点云为 $P_k$ ，将 $P_k$ 中的点投影到 $t_{k+1}$ 得到 $\bar{P}_k$ 。同时，在下一个扫描 $k+1$ 中获取的数据为 $P_{k+1}$ 。



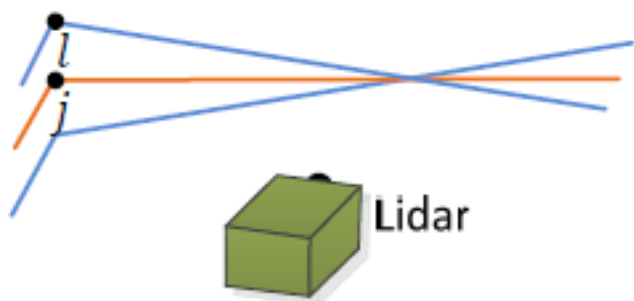
已有点集 $\bar{P}_k$ 与点集 $P_{k+1}$ ，则需要寻找两个点云数据之间的对应关系。

# LOAM算法解析

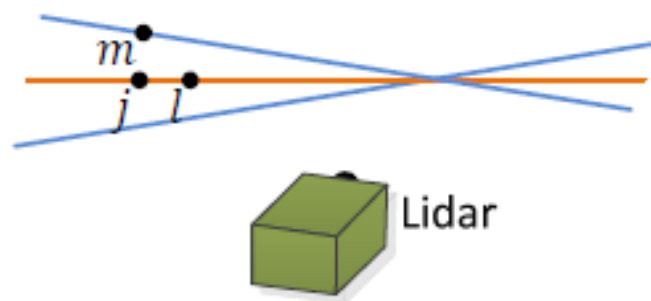
## ► Lidar Odometry

首先, 对 $P_{k+1}$ 采用前述提取特征点的方法找到两个特征点集, 边缘点 $E_{k+1}$ 和平面点 $H_{k+1}$ 。

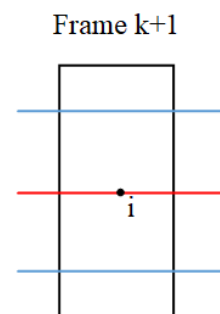
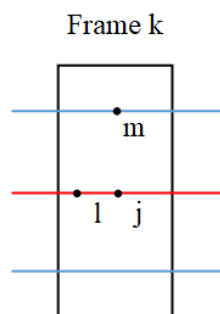
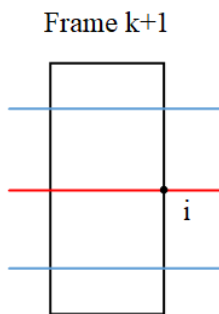
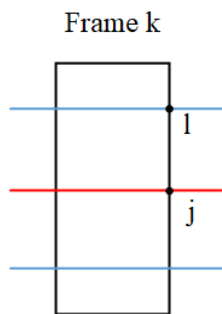
其次, 对 $\bar{P}_k$ 构建kd-tree索引, 找到 $E_{k+1}$ 中每个边缘点在 $\bar{P}_k$ 中对应的边缘线 (左图(a)) 与 $H_{k+1}$ 每个平面点在 $\bar{P}_k$ 中对应的面片 (右图(b))。



(a)



(b)





# LOAM算法解析

## ► Lidar Odometry

Step2: 计算对应距离

计算 $P_{k+1}$ 中边缘点  $i$  到 $\bar{P}_k$ 中直线 $jl$ 之间的距离:

$$d_{\mathcal{E}} = \frac{\left| (\tilde{X}_{(k+1,i)}^L - \bar{X}_{(k,j)}^L) \times (\tilde{X}_{(k+1,i)}^L - \bar{X}_{(k,l)}^L) \right|}{\left| \bar{X}_{(k,j)}^L - \bar{X}_{(k,l)}^L \right|}, \quad (2)$$

计算 $P_{k+1}$ 中平面点  $i$  到 $\bar{P}_k$ 中面片 $jlm$ 之间的距离:

$$d_{\mathcal{H}} = \frac{\left| \begin{array}{c} (\tilde{X}_{(k+1,i)}^L - \bar{X}_{(k,j)}^L) \\ ((\bar{X}_{(k,j)}^L - \bar{X}_{(k,l)}^L) \times (\bar{X}_{(k,j)}^L - \bar{X}_{(k,m)}^L)) \end{array} \right|}{\left| (\bar{X}_{(k,j)}^L - \bar{X}_{(k,l)}^L) \times (\bar{X}_{(k,j)}^L - \bar{X}_{(k,m)}^L) \right|}, \quad (3)$$

# LOAM算法解析

## ► Lidar Odometry

### Step3: 构建非线性方程组

假设LiDAR在一个扫描中角速度和线速度均为匀速运动，所以能够利用前述计算的“相对时间”对计算出的位姿进行线性插值。

$$\underline{T_{(k+1,i)}^L} = \frac{t_i - t_{k+1}}{t - t_{k+1}} T_{k+1}^L.$$

式中， $T_{k+1}^L$ 为 $[t_{k+1}, t]$ 两帧点云数据之间的位姿变换，给定一个点 $i$ ， $i \in P_{k+1}$ ，其获取的时间戳为 $t_i$ ，则 $[t_{k+1}, t_i]$ 之间的变换为 $T_{(k+1,i)}^L$ 。

每一次迭代过程中均通过下式把 $P_{k+1}$ 中的点 $i$ 转换到该帧数据的起始点 $t_{k+1}$ ，然后再与 $P_k$ 中的点进行匹配。

$$X_{(k+1,i)}^L = \mathbf{R} \tilde{X}_{(k+1,i)}^L + T_{(k+1,i)}^L(1:3),$$

组合 $P_{k+1}$ 中边缘点到 $\bar{P}_k$ 中边缘线，平面点到面片之间的距离方程：

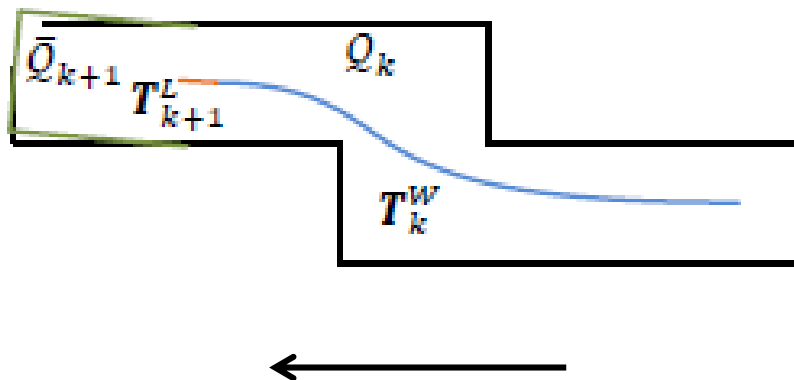
$$\begin{aligned} f_{\mathcal{E}}(X_{(k+1,i)}^L, T_{k+1}^L) &= d_{\mathcal{E}}, \quad i \in \mathcal{E}_{k+1}. \\ f_{\mathcal{H}}(X_{(k+1,i)}^L, T_{k+1}^L) &= d_{\mathcal{H}}, \quad i \in \mathcal{H}_{k+1}. \end{aligned} \quad \longrightarrow \quad f(T_{k+1}^L) = d,$$

# LOAM算法解析

## ► Lidar Mapping

Lidar Odometry 高频、低精度运行，作者在Mapping模块对其输出的位姿参数进一步优化。

Mapping模块在每个扫描结束才运行一次，如在第 $k+1$ 次扫描结束后，lidar odometry产生了一个没有畸变的点云 $\bar{P}_{k+1}$ 和一个位姿变换参数 $T_{k+1}^L$ （代表 $[t_{k+1}, t_{k+2}]$ 之间的运动）。如下图中所示，假设 $Q_k$ 为前 $k$ 次扫描积累的数据所形成的点云地图， $T_k^W$ 为lidar在地图中最后一次扫描 $k$ 到 $k+1$ 处的变换位姿。随着lidar odometry的输出，Mapping算法不断扩展 $T_k^W$ 为 $T_{k+1}^W$ （即扫描时间 $[t_{k+1}, t_{k+2}]$ 范围的全局变换，并把 $\bar{P}_{k+1}$ 投影到全局坐标系 $W$ 下，得到 $\bar{Q}_{k+1}$ 。接着，通过优化 $T_{k+1}^W$ 把 $\bar{Q}_{k+1}$ 匹配到地图 $Q_k$ 中。



**谢谢！**

