深蓝学院
shenlanxueyuan.com

多传感器融合定位

第2讲 3D激光里程计 I

主讲人 任 乾

北京理工大学本硕
自动驾驶从业者

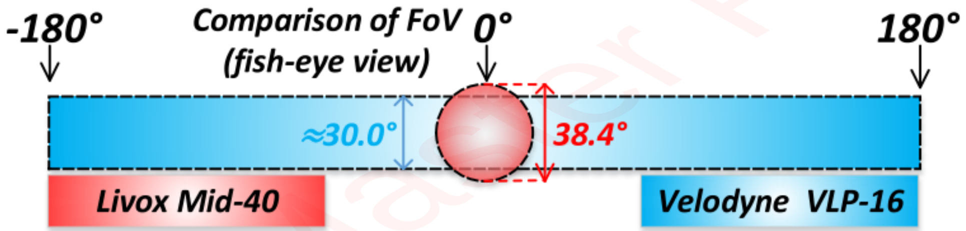
目录

激光传感器原理

Lidar的分类：机械旋转激光雷达(如vlp16)，固态激光雷达(如Livox Mid-40)

不同点：

a. 视角范围不相同



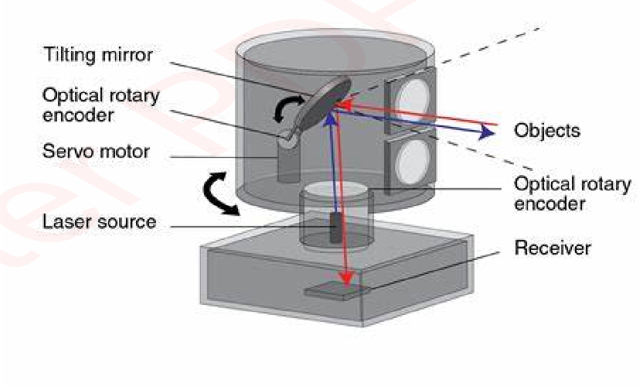
b. 扫描工作方式不同

机械旋转激光雷达: 一般是多个激光束同时发射，并绕固定轴旋转，来探测三维环境
缺点：远处的激光点之间的间隔较大

激光传感器原理

机械Lidar的工作方式：

- 激光雷达传感器向周围环境发射脉冲光波；
- 这些脉冲碰撞到周围物体反弹并返回传感器；
- 传感器使用每个脉冲返回到传感器所花费的时间来计算其传播的距离；
- 每秒重复数百万次此过程，将创建精确的实时3D环境地图。

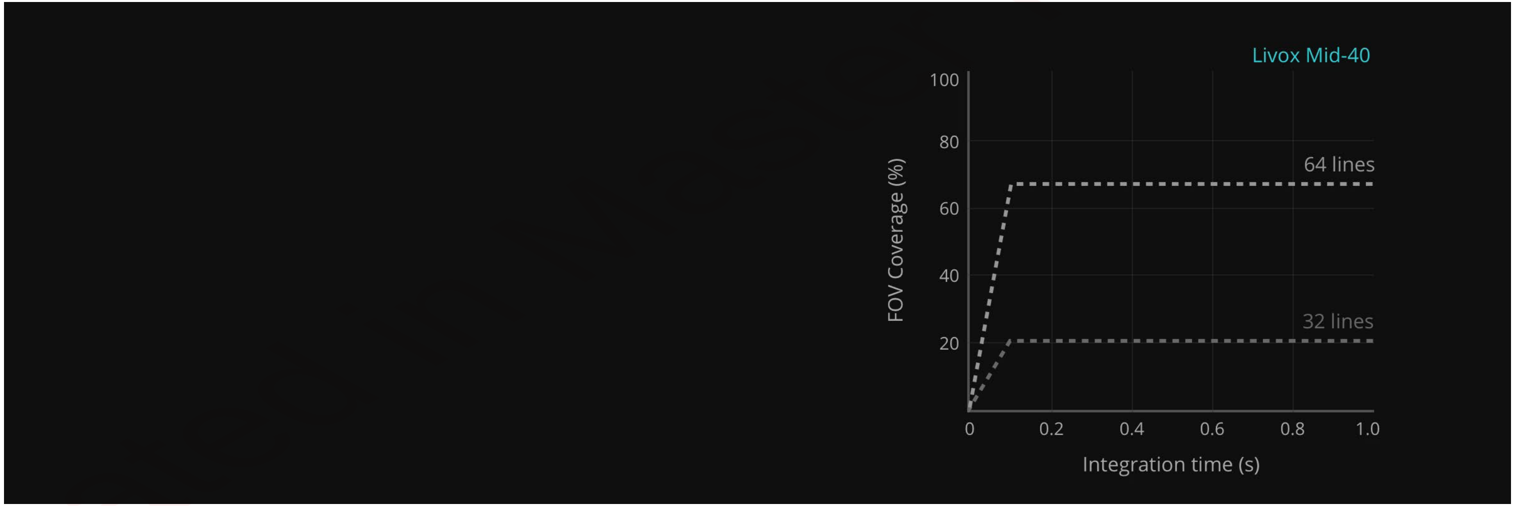


参考视频：第一章第一节 激光雷达点云采集原理（以Velodyne Lidar为例）

激光传感器原理

固态激光雷达：非重复扫描，它的扫描方式是梅花瓣状的。

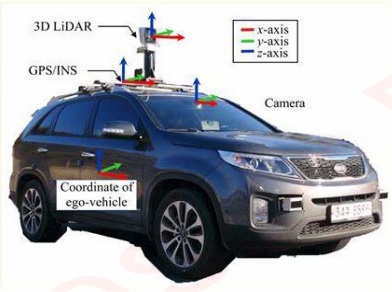
下图展示了Livox Mid-40静态扫描的激光点在$X = 1$平面上的点投影，我们可以看到每一帧激光点并不重合，这意味着在Livox Mid-40在静态放置的状态下，经过多次扫描可以获得比机械旋转激光更稠密的点云。

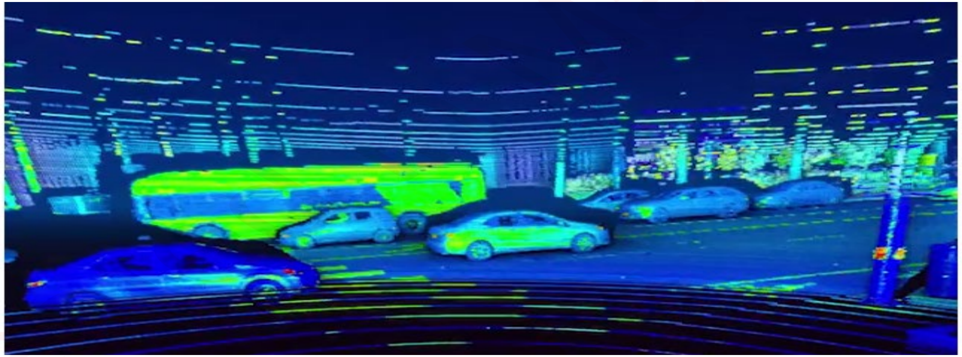


开源代码：https://github.com/hku-mars/loam_livox

激光传感器原理

Lidar的应用：



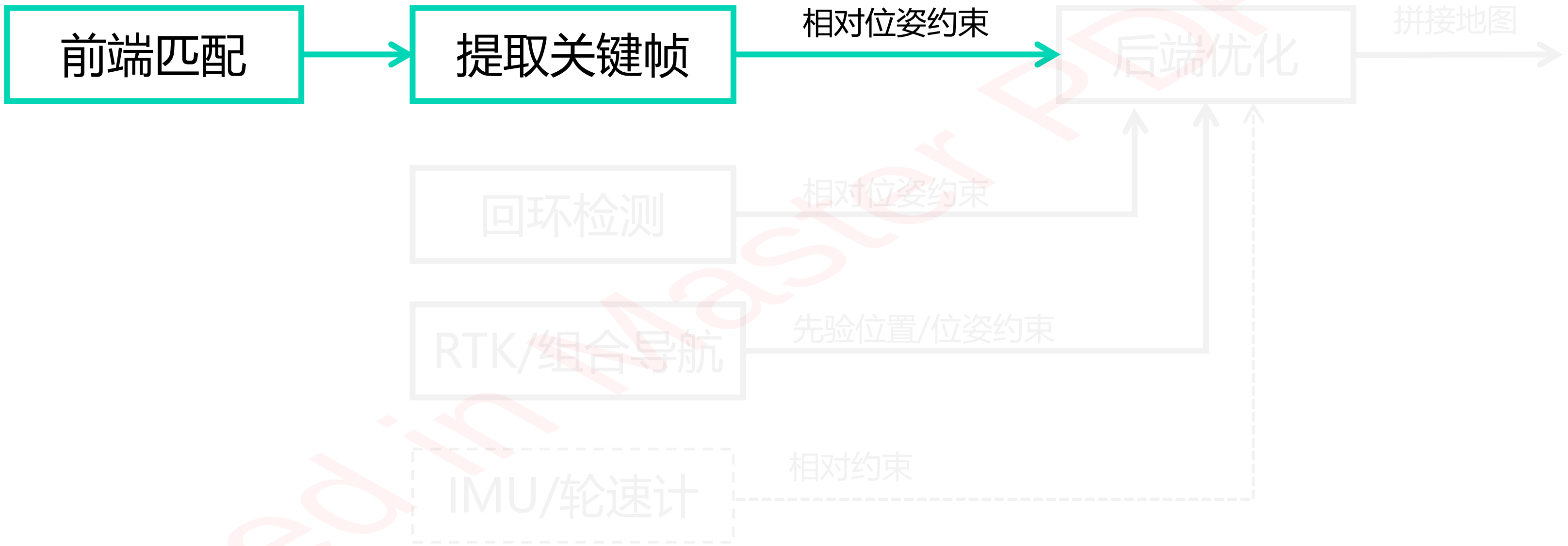


目录

整体流程介绍

点云地图构建流程

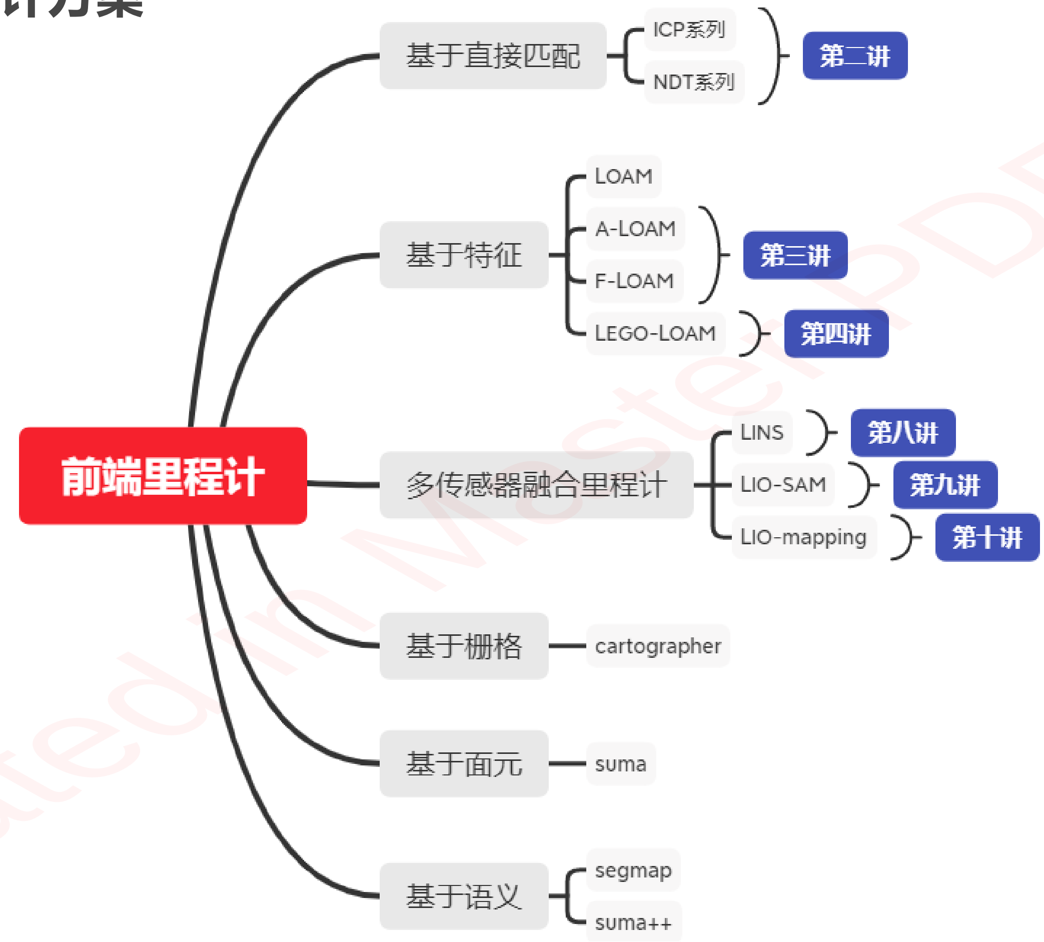
整体流程介绍



点云地图构建流程

目录

前端里程计方案



前端里程计
基于直接匹配
ICP系列
NDT系列
第二讲
基于特征
LOAM
A-LOAM
F-LOAM
第三讲
LEGO-LOAM
第四讲
多传感器融合里程计
LINS
第八讲
LIO-SAM
第九讲
LIO-mapping
第十讲
基于栅格
cartographer
基于面元
suma
基于语义
segmap
suma++

前端里程计方案—基于直接匹配

点到点ICP-基于SVD

点集：

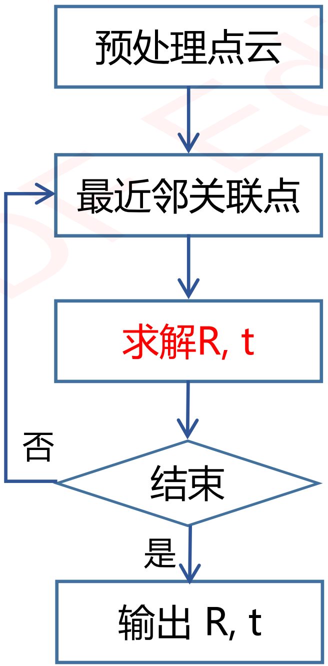
$$X = \{x_1, x_2, \cdots, x_{N_x}\}$$

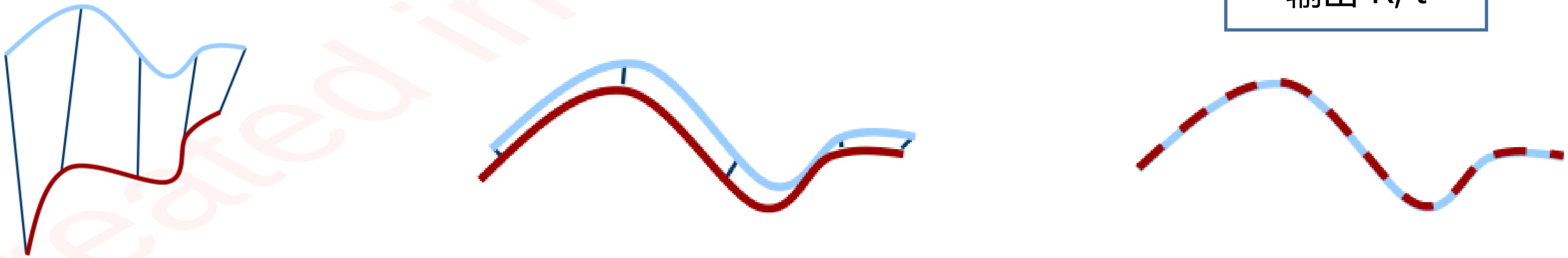
$$Y = \{y_1, y_2, \cdots, y_{N_y}\}$$

其中 X 和 Y 是原始点云的子集，选取的是两个点集中能够互相关联的那些点，即 $N_x = N_y$

目标：

$$minE(R,t) = min\frac{1}{N_y}\sum_{i=1}^{N_y}\|x_i - Ry_i - t\|^2$$





前端里程计方案—基于直接匹配

点到点ICP-基于SVD

$$
\begin{aligned}
E(R,t) &= \frac{1}{N_y}\sum_{i=1}^{N_y}\|x_i - Ry_i - t - u_x + Ru_y + u_x - Ru_y\|^2 \\
&= \frac{1}{N_y}\sum_{i=1}^{N_y}\left(\|x_i - u_x - R\left(y_i - u_y\right) + \left(u_x - Ru_y - t\right)\|^2\right) \\
&= \frac{1}{N_y}\sum_{i=1}^{N_y}\left(\|x_i - u_x - R\left(y_i - u_y\right)\|^2 + \|u_x - Ru_y - t\|^2 + 2\boxed{\left(x_i - u_x - R\left(y_i - u_y\right)\right)}^T\left(u_x - Ru_y - t\right)\right) \\
&= \frac{1}{N_y}\sum_{i=1}^{N_y}\left(\|x_i - u_x - R\left(y_i - u_y\right)\|^2 + \|u_x - Ru_y - t\|^2\right)
\end{aligned}
$$

由 u_x 和 u_y 的定义可知，
该项累加和为零

其中 u_x 和 u_y 分别是点集 X 和 Y 的质心，即

$$
u_x = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i \qquad u_y = \frac{1}{N_y}\sum_{i=1}^{N_y} y_i
$$

前端里程计方案—基于直接匹配

点到点ICP-基于SVD

$$E(R,t)=\frac{1}{N_y}\sum_{i=1}^{N_y}\left(\|x_i-u_x-R\left(y_i-u_y\right)\|^2+\|u_x-Ru_y-t\|^2\right)$$

令 $E_1(R,t)=\frac{1}{N_y}\sum_{i=1}^{N_y}\|x_i-u_x-R\left(y_i-u_y\right)\|^2$ (只与旋转有关)

$E_2(R,t)=\|u_x-Ru_y-t\|^2$ (用于求平移部分)

则 $E(R,t)=E_1(R,t)+E_2(R,t)$

那么，对于任意的R，均可以找到一个t，使得$u_x-Ru_y-t=0$，即$E_2(R,t)=0$，

因此，可以先根据$E_1(R,t)$求旋转，再根据$E_2(R,t)$求平移。

前端里程计方案—基于直接匹配

点到点ICP-基于SVD

$$
\begin{aligned}
E_1(R,t) &= \frac{1}{N_y}\sum_{i=1}^{N_y}\left\|x_i - u_x - R\left(y_i - u_y\right)\right\|^2 \\
&= \frac{1}{N_y}\sum_{i=1}^{N_y}\left\|x_i' - Ry_i'\right\|^2 \\
&= \frac{1}{N_y}\sum_{i=1}^{N_y}\left(\boxed{x_i'^T x_i'} + y_i'^T \boxed{R^T R} y_i' - 2x_i'^T R y_i'\right)
\end{aligned}
$$

与R无关　单位阵

令 $E_1'(R,t) = \sum_{i=1}^{N_y} x_i'^T R y_i'$

则

$$
\begin{aligned}
&\arg\min_R E_1(R,t) \\
=&\arg\max_R E_1'(R,t) \\
=&\arg\max_R \sum_{i=1}^{N_y} x_i'^T R y_i'
\end{aligned}
$$

$$
E_1'(R,t) = \sum_{i=1}^{N_y} x_i'^T R y_i' \overset{(1)}{=} \sum_{i=1}^{N_y}\operatorname{Trace}\left(x_i'^T R y_i'\right) \overset{(2)}{=} \sum_{i=1}^{N_y}\operatorname{Trace}\left(R y_i' x_i'^T\right) = \operatorname{Trace}\left(\sum_{i=1}^{N_y} R y_i' x_i'^T\right) \overset{(3)}{=} \operatorname{Trace}(RH)
$$

等式(1)：标量的迹等于它自身

等式(3)：$H = \sum_{i=1}^{N_y} y_i' x_i'^T$

等式(2)：　$\operatorname{tr}(AB) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}b_{ji} = \sum_{i=1}^{m}\sum_{j=1}^{n} b_{ji}a_{ij} = \operatorname{tr}(BA)$

问题转化为，找到合适的R，使 $\operatorname{Trace}(RH)$ 达到最大值

前端里程计方案—基于直接匹配

点到点ICP-基于SVD

1. 旋转部分求解

定理：若有正定矩阵 AA^T，则对于任意正交矩阵 B，有 $\mathrm{Trace}(AA^T) \geq \mathrm{Trace}(BAA^T)$

意义：若能找到 R，把 $\mathrm{Trace}(RH)$ 转换成 $\mathrm{Trace}\left(AA^T\right)$ 的形式，则该 R 就是我们要找的旋转矩阵

证明：

$$\operatorname{tr}\left(BAA^T\right) = \operatorname{tr}\left(A^T BA\right) = \sum_i a_i^T \left(Ba_i\right)$$

其中 a_i 为 A 的列向量。根据柯西-施瓦茨不等式，有

$$a_i^T \left(Ba_i\right) \leq \sqrt{\left(a_i^T a_i\right)\left(a_i^T B^T B a_i\right)} = a_i^T a_i$$

因此

$$\operatorname{tr}\left(BAA^T\right) = \sum_i a_i^T \left(Ba_i\right) \leq \sum_i a_i^T a_i = \operatorname{tr}\left(AA^T\right)$$

前端里程计方案—基于直接匹配

点到点ICP-基于SVD

目的：找到 R ，把 $\operatorname{Trace}(RH)$ 转换成 $\operatorname{Trace}\left(AA^T\right)$ 的形式

方法：

对H进行SVD分解

$$H = U\Sigma V^T$$

取

$$R = VU^T$$

则有

$$RH = VU^T U\Sigma V^T = V\Sigma V^T = V\Sigma^{\frac{1}{2}}\Sigma^{\frac{1}{2}}V^T = V\Sigma^{\frac{1}{2}}\left(V\Sigma^{\frac{1}{2}}\right)^T$$

前端里程计方案—基于直接匹配

点到点ICP-基于SVD

2. 平移部分求解

旋转 R 确定之后，可根据

$$E_2(R,t) = \|u_x - Ru_y - t\|^2$$

直接得到平移量为

$$t = u_x - Ru_y$$

参考文献：

1) Arun, K. Somani, Thomas S. Huang, and Steven D. Blostein. "Least-squares fitting of two 3-D point sets." IEEE Transactions on pattern analysis and machine intelligence 5 (1987): 698-700.

2) https://igl.ethz.ch/projects/ARAP/svd_rot.pdf

前端里程计方案—基于直接匹配

凸优化基础

1. 优化任务的目标

找到 n 维的变量 $x^* \in \mathbb{R}^n$，使得损失函数 $F(x)$ 取局部最小值：

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

局部最小值指对任意的 $\|x - x^*\| < \delta$，都有 $F(x^*) \leq F(x)$

方程中，$f(x)$ 是残差函数，在实际使用中，它可以代表任何方式得到的残差，比如 ICP 中点到点之间的距离、融合中预测与观测之间的误差等等。

2. 迭代方法的思路

当方程形式复杂时，无法直接求出解析解，因此需要使用迭代方法，找到最优解，步骤为：

1) 给定某个初值 x_0；

2) 对于第k次迭代，寻找增量 Δx_k，使得 $\|f(x_k + \Delta x_k)\|_2^2$ 达到极小值；

3) 若 Δx_k 足够小，则停止；

4) 否则，令 $x_{k+1} = x_k + \Delta x_k$，返回第2步。

前端里程计方案—基于直接匹配

凸优化基础

3. 迭代下降求解优化问题的方法

损失函数可以泰勒展开如下

$$F(x+\Delta x) \approx F(x) + J\Delta x + \frac{1}{2}\Delta x^{\top} H \Delta x$$

其中 J 和 H 分别为损失函数 F 对变量 X 的一阶导(也叫梯度或Jacobian矩阵)和二阶导(也叫Hessian矩阵)。

3.1 最速下降法

1) 原理

只保留一阶泰勒展开结果，取增量为

$$\Delta x^* = -J^T$$

即沿梯度的反方向取增量，则可以保证使损失函数减小。

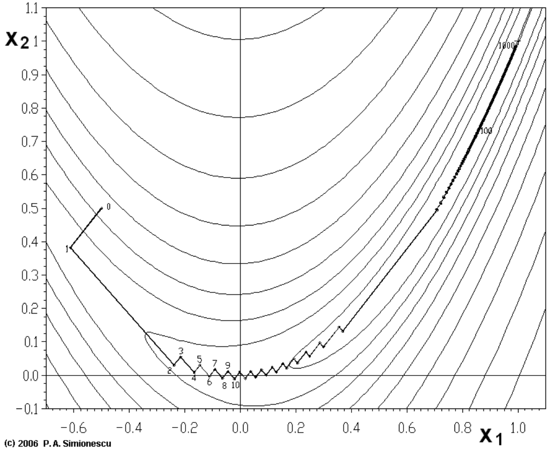
2) 优点

迭代方便、计算简单

3) 缺点

a. 一阶近似精度有限，容易走出锯齿形状

b. 越接近目标值，步长越小，前进越慢



凸优化基础

3. 迭代下降求解优化问题的方法

3.2 牛顿法

1) 原理

保留二阶泰勒展开结果，此时的增量方程为

$$\Delta x^{*}=\arg\min\left(F(x)+J\Delta x+\frac{1}{2}\Delta x^{\mathrm{T}}H\Delta x\right)$$

求右侧等式关于Δx的导数，并令它为零

$$J^{T}+H\Delta x=0\Rightarrow H\Delta x=-J^{T}$$

求解该方程，即可得到所需的增量。

2) 优点

a. 对原函数的近似更精确，每一步的收敛更加准确

b. 收敛速度快

3) 缺点

需要计算 H 矩阵，在优化规模较大时，不容易做到。

凸优化基础

3.3 高斯牛顿法

1) 原理

对 $f(x)$ 进行泰勒展开，而非 $F(x)$

$$f(x+\Delta x) \approx f(x) + J\Delta x$$

此时优化问题变成了，寻找增量 Δx ，使得

$\|f(x+\Delta x)\|^2$ 达到最小，即

$$\Delta x^* = \arg\min_{\Delta x} \frac{1}{2}\|f(x) + J\Delta x\|^2$$

同样的，需要对右侧求导，并令导数为零。

右侧展开为

$$\begin{aligned}
&\frac{1}{2}\|f(x) + J\Delta x\|^2 \\
=&\frac{1}{2}(f(x) + J\Delta x)^{\mathrm{T}}(f(x) + J\Delta x) \\
=&\frac{1}{2}\left(\|f(x)\|_2^2 + 2f(x)J\Delta x + \Delta x^{\mathrm{T}} J^T J\Delta x\right)
\end{aligned}$$

求导并令导数为零

$$J^T f(x) + J^T J\Delta x = 0$$

即

$$\underbrace{J^T J}_{H}\Delta x = \underbrace{-J^T f(x)}_{g}$$

凸优化基础

最终的求解的增量为

$$\Delta x = H^{-1} g$$

2) 优点

用 $J^T J$ 做为牛顿法中 H 的近似，从而避免了直接求解二阶导数矩阵。

3) 缺点

求解增量就必须保证 H 矩阵是可逆的，而 $J^T J$ 只能保证半正定，此时算法稳定性变差，最终导致不收敛。

3.4 LM方法

LM的原理推导较为复杂，此处直接给出增量方程形式：

$$(H + \lambda I)\Delta x = g$$

该方法好处是可一定程度避免 H 不正定带来的病态问题。

实际使用中，当问题性质较好时，用高斯牛顿法；问题接近病态时，用LM方法。

参考文献：
1) 《视觉SLAM十四讲》第6.2节.
2) http://www2.imm.dtu.dk/pubdb/edoc/imm3215.pdf

前端里程计方案—基于直接匹配

点到点ICP-基于优化

非线性优化要求雅可比，而基于优化的ICP对应的雅可比推导在李代数模式下会更加简洁，李代数基础等到后面章节才介绍，因此此处直接给出原理和结论，相关细节在后续章节给出。

ICP求解问题，可以重新表示为

$$\min_{T} = \frac{1}{2}\sum_{i=1}^{n}\|(x_i - Ty_i)\|_2^2$$

其中 T 为位姿变换矩阵

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

在李代数模式下，又可以重新表示为

$$\min_{\xi} = \frac{1}{2}\sum_{i=1}^{n}\|(x_i - \exp(\xi^{\wedge})\, y_i)\|_2^2$$

其中 ξ 为 T 对应的李代数

残差对应的雅可比为

$$J = \frac{\partial e}{\partial \delta\xi} = -\left(\exp(\xi^{\wedge})\, y_i\right)^{\odot}$$

随后，便可根据优化的固定步骤求解位姿。

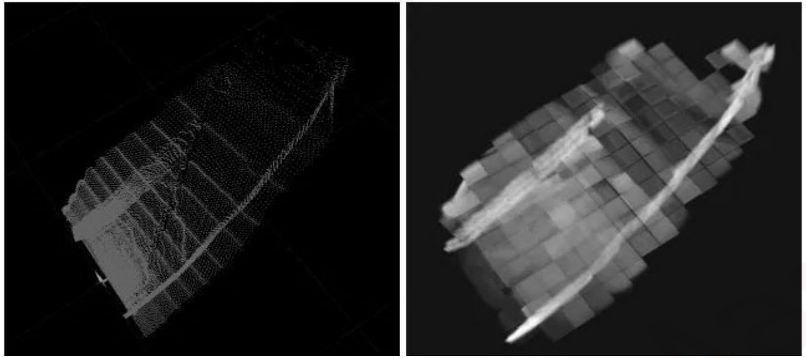
参考文献：
1) 《机器人学中的状态估计》第8.1.3节

前端里程计方案—基于直接匹配

ICP系列汇总

前端里程计方案—基于直接匹配

NDT系列—经典NDT



点集：

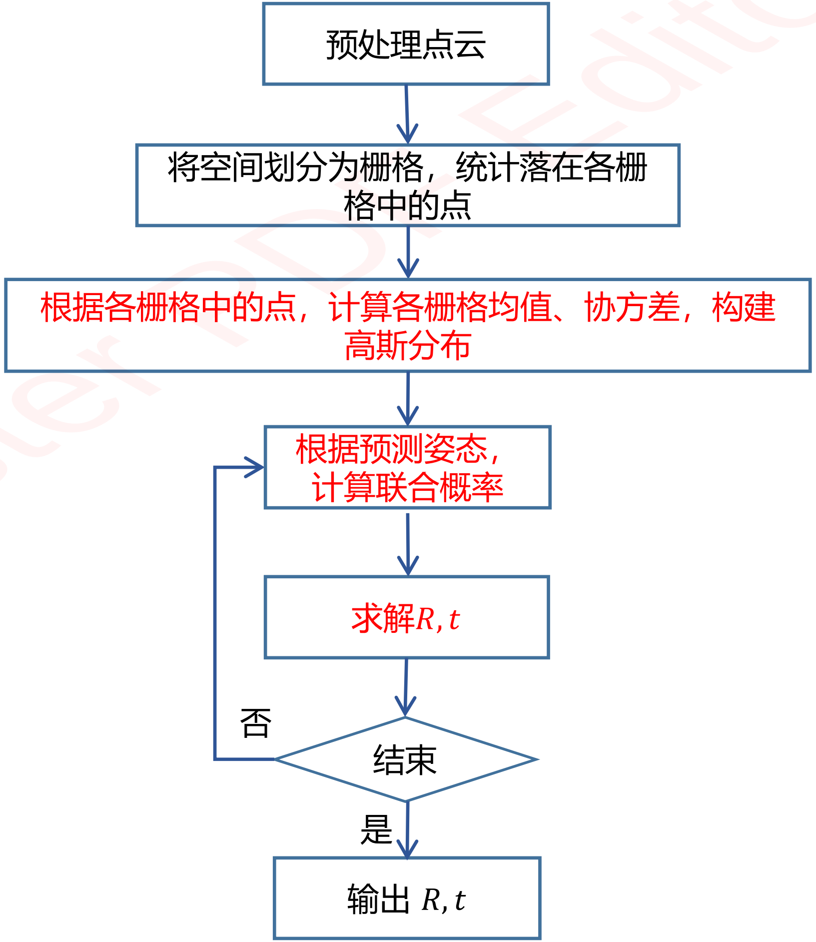
$$X=\{x_1,x_2,\cdots,x_{N_x}\}$$

$$Y=\{y_1,y_2,\cdots,y_{N_y}\}$$

目标： $max\Psi = max\prod_{i=1}^{N_y} f\left(X, T\left(p, y_i\right)\right)$

2D模型： $p=p_3=[t_x \quad t_y \quad \phi_z]^{\mathrm{T}}$

3D模型： $p=p_6=[t_x \quad t_y \quad t_z \quad \phi_x \quad \phi_y \quad \phi_z]^{\mathrm{T}}$



前端里程计方案—基于直接匹配

NDT系列—经典NDT

均值：

$$\mu = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i$$

协方差：

$$\mathbf{\Sigma} = \frac{1}{N_x - 1}\sum_{i=1}^{N_x}(x_i - \mu)(x_i - \mu)^{\mathrm{T}}$$

根据预测的位姿，对点进行旋转和平移：

$$y_i' = T(p, y_i) = Ry_i + t$$

旋转和平移后的点与目标点集中的点在同一
坐标系下，此时可计算各点的联合概率：

$$f(X, y_i') = \frac{1}{\sqrt{2\pi}\sqrt{|\mathbf{\Sigma}|}}\exp\left(-\frac{(y_i' - \mu)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(y_i' - \mu)}{2}\right)$$

所有点的联合概率：

$$\Psi = \prod_{i=1}^{N_y} f(X, T(p, y_i))$$

$$= \prod_{i=1}^{N_y}\frac{1}{\sqrt{2\pi}\sqrt{|\mathbf{\Sigma}|}}\exp\left(-\frac{(y_i' - \mu)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(y_i' - \mu)}{2}\right)$$

取对数，简化问题：

$$ln\Psi = \sum_{i=1}^{N_y}\left(-\frac{(y_i' - \mu)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(y_i' - \mu)}{2} + \boxed{ln\left(\frac{1}{\sqrt{2\pi}\sqrt{|\mathbf{\Sigma}|}}\right)}\right)$$

常数

去除常数项：

$$max\Psi = maxln\Psi = min\Psi_1 = min\sum_{i=1}^{N_y}(y_i' - \mu)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(y_i' - \mu)$$

前端里程计方案—基于直接匹配

NDT系列—经典NDT

目标函数：$min \sum_{i=1}^{N_y} (y_i' - \mu)^{\mathrm{T}} \mathbf{\Sigma}^{-1} (y_i' - \mu)$

$$y_i' = T(p, y_i) = Ry_i + t$$

待求参数：R, t

why can we have this instead of
sigma^(-1/2)(yi-mu)?

定义残差函数

$$f_i(p) = y_i' - \mu$$

按照高斯牛顿法的流程，只需计算残差函数关于待求参数的雅可比，便可迭代优化。

$$J_i = \frac{df_i(p)}{dp}$$

2D场景求解：

$$p = [t_x \quad t_y \quad \phi_z]^{\mathrm{T}}$$

$$y_i' = T(p, y_i)$$
$$= \begin{bmatrix} \cos\phi_z & -\sin\phi_z \\ \sin\phi_z & \cos\phi_z \end{bmatrix} y_i + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

雅可比 $J_i = \begin{bmatrix} 1 & 0 & -y_{i1}\sin\phi_z - y_{i2}\cos\phi_z \\ 0 & 1 & y_{i1}\cos\phi_z - y_{i2}\sin\phi_z \end{bmatrix}$

前端里程计方案—基于直接匹配

NDT系列—经典NDT

3D场景求解：

$$p = [t_x \quad t_y \quad t_z \quad \phi_x \quad \phi_y \quad \phi_z]^{\mathrm{T}}$$

$$y_i' = T(p, y_i) = R_x R_y R_z y_i + t = \begin{bmatrix} c_y c_z & -c_y s_z & s_y \\ c_x s_z + s_x s_y c_z & c_x c_z - s_x s_y s_z & -s_x c_y \\ s_x s_z - c_x s_y c_z & c_x s_y s_z + s_x c_z & c_x c_y \end{bmatrix} y_i + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

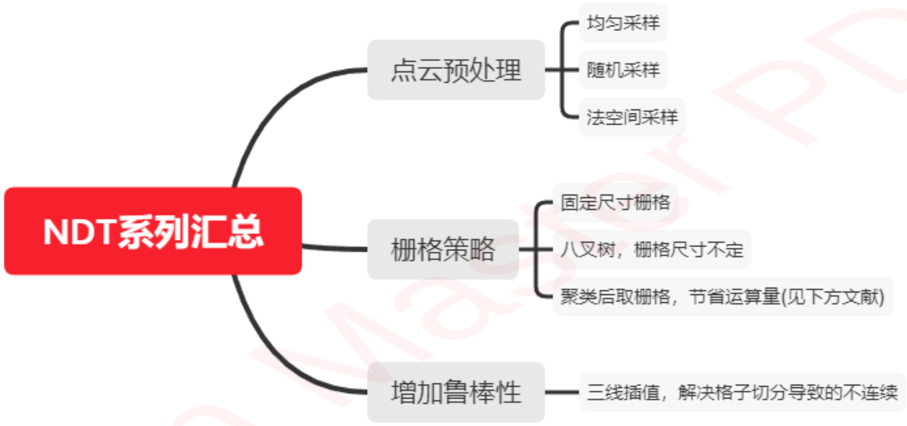
雅可比 $J_i = \begin{bmatrix} 1 & 0 & 0 & 0 & c & f \\ 0 & 1 & 0 & a & d & g \\ 0 & 0 & 1 & b & e & h \end{bmatrix}$

其中

$$\begin{aligned}
a &= y_{i1}(-s_x s_z + c_x s_y c_z) + y_{i2}(-s_x c_z - c_x s_y s_z) + y_{i3}(-c_x c_y) \\
b &= y_{i1}(c_x s_z + s_x s_y c_z) + y_{i2}(-s_x s_y s_z + c_x c_z) + y_{i3}(-s_x c_y) \\
c &= y_{i1}(-s_y c_z) + y_{i2}(s_y s_z) + y_{i3}(c_y); \\
d &= y_{i1}(s_x c_y c_z) + y_{i2}(-s_x c_y s_z) + y_{i3}(s_x s_y) \\
e &= y_{i1}(-c_x c_y c_z) + y_{i2}(c_x c_y s_z) + y_{i3}(-c_x s_y); \\
f &= y_{i1}(-c_y s_z) + y_{i2}(-c_y c_z) \\
g &= y_{i1}(c_x c_z - s_x s_y s_z) + y_{i2}(-c_x s_z - s_x s_y c_z); \\
h &= y_{i1}(s_x c_z + c_x s_y s_z) + y_{i2}(c_x s_y c_z - s_x s_z)
\end{aligned}$$

前端里程计方案—基于直接匹配

NDT系列—其他NDT



1. Scan Registration using Segmented Region Growing NDT. Das A, Waslander SL. 2014.

2. 3D Scan Registration Using the Normal Distributions Transform with Ground Segmentation and Point Cloud Clustering. Das A, Waslander SL. 2013.

3. Scan Registration with Multi-Scale K-Means Normal Distributions Transform. Das A, Waslander SL. 2012.

目录

点云畸变补偿

1. 产生原因

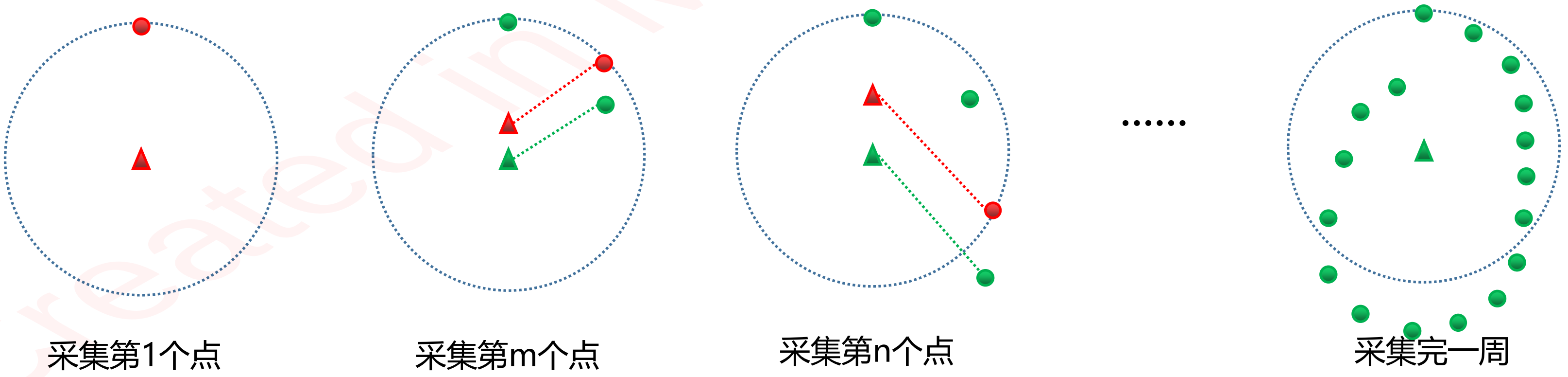
一帧点云：通常指雷达内部旋转一周扫描得到的点的集合

优点：有足够数量的点云才能进行匹配，且一周正好是周围环境的完整采集。

缺点：每个激光点的坐标都是相对于雷达的，雷达运动时，不同激光点的坐标原点会不同

1) 平移导致的畸变

虚线圆圈为真实物体；**红色点**为激光束打到的位置；**红色三角**为当前采集时刻雷达的位置；**绿色三角**为一帧的坐标原点；**绿色点**为一帧点云中激光点的坐标

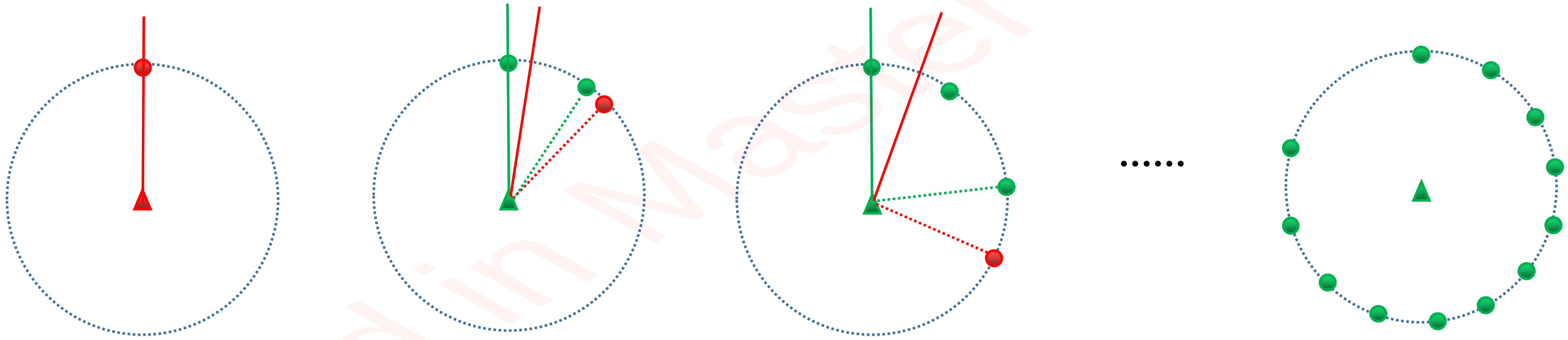


点云畸变补偿

2) 旋转导致的畸变

假设雷达在顺时针旋转

红色实线为雷达0度坐标轴，**绿色实线**为一帧点云0度坐标轴



采集第1个点　采集第m个点　采集第n个点　采集完一周

点云畸变补偿

2. 补偿方法

对每个激光点坐标做补偿，补偿量为激光点原点(即当时雷达坐标)相对于该帧起始时刻的变化。

假设一帧点云中，起始时刻雷达的位姿为

$$T_0 = \begin{bmatrix} R_0 & t_0 \\ 0 & 1 \end{bmatrix}$$

第i个激光点采集时，雷达的位姿为

$$T_i = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix}$$

第i个激光点的坐标为

$$P_i = \begin{bmatrix} p_{ix} & p_{iy} & p_{iz} \end{bmatrix}^T$$

则第i个激光点补偿畸变后的坐标应该为

$$\bar{P}_i = T_0^{-1} T_i P_i$$

上式可以理解为，只需要计算0到i时刻，激光雷达的相对旋转和相对平移变化即可。

实际上，雷达点云是局部坐标系下的表示，当以0时刻雷达的位姿为基准坐标系时，此时T_0即为单位阵，T_i即为0到i时刻的相对旋转和平移。

此时有

$$R_i = w d_{time}$$
$$t_i = V d_{time}$$

即，只需要知道0到i时刻的平均角速度和平均速度即可。

点云畸变补偿

2. 补偿方法

```
adjusted_cloud_ptr->points.resize(input_cloud_ptr->points.size());
Eigen::Vector3f point, rotated_point, adjusted_point;
Eigen::Matrix3f current_matrix;
bool half_passed = false;
for (size_t point_index = 0; point_index < input_cloud_ptr->points.size();
     point_index++) {
  point(0) = input_cloud_ptr->points[point_index].x;
  point(1) = input_cloud_ptr->points[point_index].y;
  point(2) = input_cloud_ptr->points[point_index].z;
  float ori = -atan2(point(1), point(0));
  if (!half_passed) {
    if (ori < start_orientation - M_PI / 2)
      ori += 2 * M_PI;
    else if (ori > start_orientation + M_PI * 3 / 2)
      ori -= 2 * M_PI;
    if (ori - start_orientation > M_PI) half_passed = true;
  } else {
    ori += 2 * M_PI;
    if (ori < end_orientation - M_PI * 3 / 2)
      ori += 2 * M_PI;
    else if (ori > end_orientation + M_PI / 2)
      ori -= 2 * M_PI;
  }
  float real_time =
      (ori - start_orientation) / orientation_diff * scan_period_;
  current_matrix = UpdateMatrix(real_time);
  rotated_point = current_matrix * point;
  adjusted_point = rotated_point + velocity_ * real_time;
  adjusted_cloud_ptr->points[point_index].x = adjusted_point(0);
  adjusted_cloud_ptr->points[point_index].y = adjusted_point(1);
  adjusted_cloud_ptr->points[point_index].z = adjusted_point(2);
}
```

使用方法：

1）正常雷达驱动输出的数据(如velodyne驱动)均可按此逻辑进行补偿。

2）角速度和线速度输入，可以使用imu、编码器等外接传感器，也可以使用slam的相对位姿，后者效果会稍差。

特别事项：

作业工程中提供的畸变补偿方法与ppt中提供的代码不同，原因是kitti提供的数据打乱了点云排列格式。

由于这种问题在实际工程中不会出现，且该方法理解较为复杂，故此处只对思路做大致介绍，各位可跳过对作业工程中该部分代码的学习。

点云畸变补偿

3. 进阶思考

1）上述方法以一帧点云起始时间作为该帧点云时间，若以终止时间或中间时刻作为点云时间，畸变补偿方法应如何变化?

2) 若雷达内部是逆时针旋转，而不是顺时针旋转，畸变补偿方法应如何变化?

若想对各种情况变化做出适应性改动，须理解雷达运动造成点云畸变的核心机理才行，不要死记方法。

目录

基于数据集实现

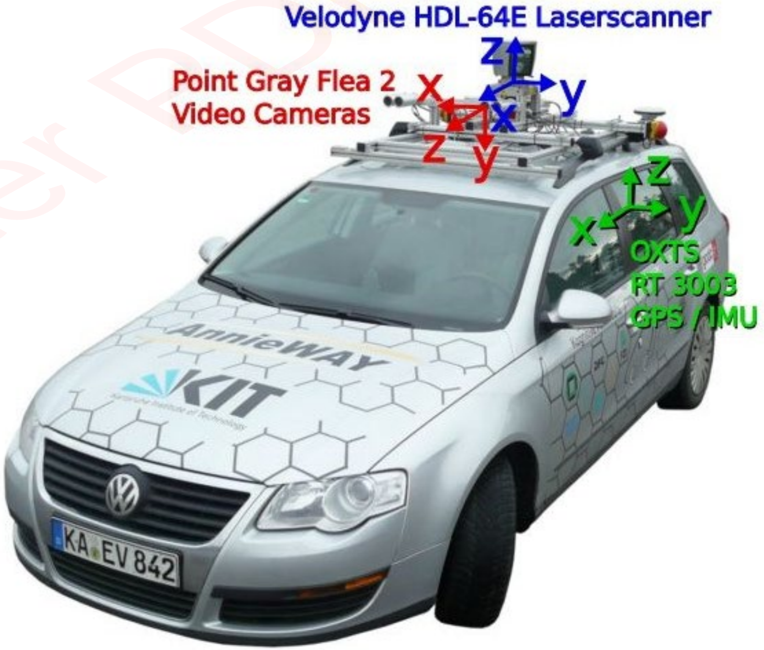
1. KITTI数据集简介

硬件组成：

1) 一个64线激光雷达，在车顶的正中心

2) 两个彩色摄像头和两个黑白摄像头，在雷达两侧。

3) 一个组合导航系统（OXTS RT 3003），在雷达左后方。它可以输出RTK/IMU组合导航结果，包括经纬度和姿态，同时也输出IMU原始数据。

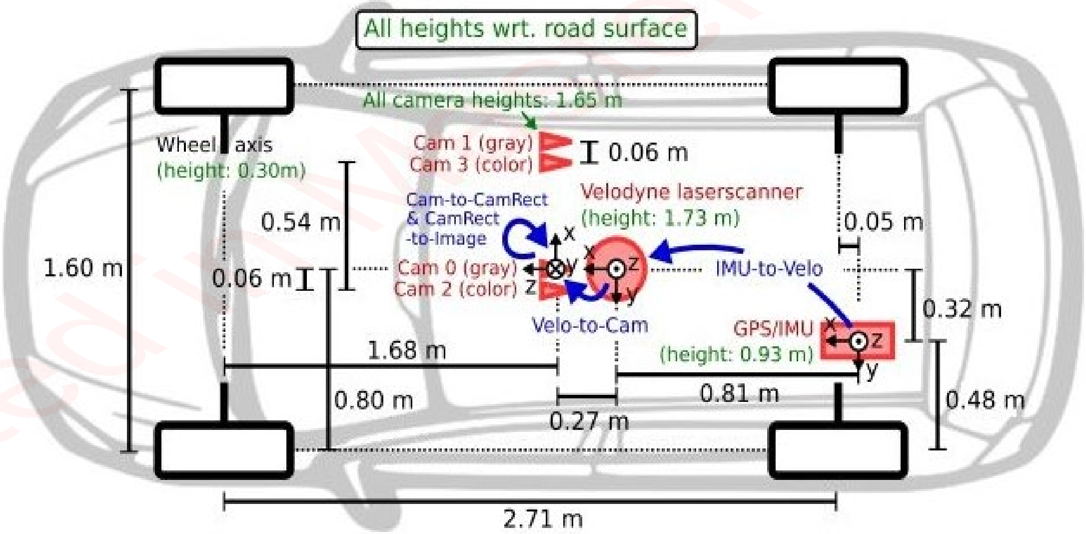


基于数据集实现

1. KITTI数据集简介

安装关系：

图中所示的所有安装关系，都可以在数据集提供标定文件中找到，可直接使用。



基于数据集实现

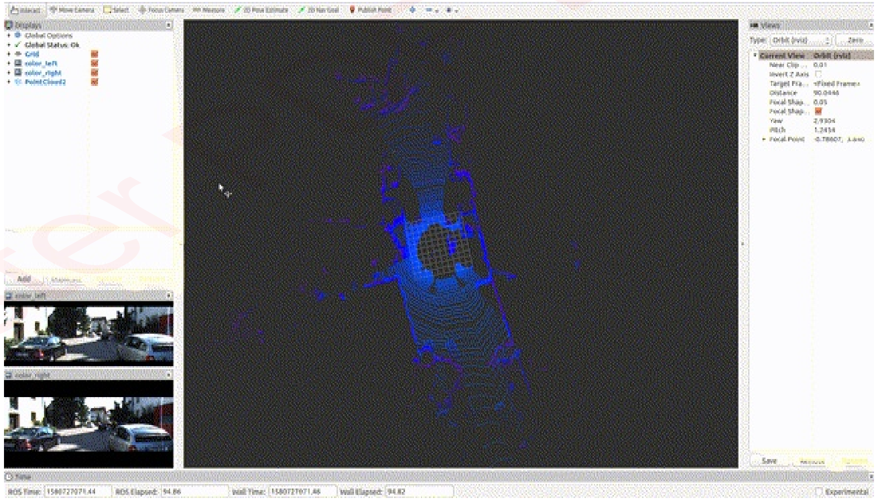
2. 数据集及使用

1) 下载数据集

由于kitti原始数据有问题，建议直接下载课程提供的bag文件(地址：https://share.weiyun.com/GxqcTaE2)

2) 测试bag

a. roscore
b. rviz -d (需要先 cd 到 rviz 文件所在目录下)
c. rosbag play kitti_2011_10_03_drive_0027_synced.bag



基于数据集实现

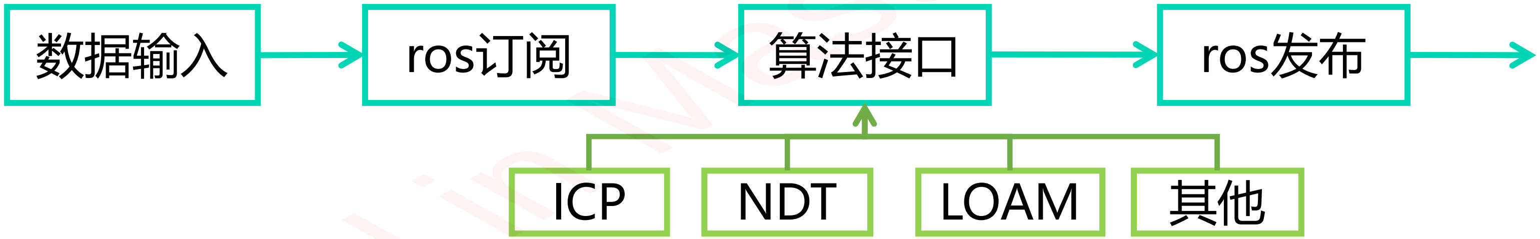
3. 里程计工程框架实现

核心思想：

1) 通过类的封装，实现模块化

2) 把ros流程与c++内部实现分开，使流程清晰

3) 基于c++多态，实现高可扩展性。



参考文章：

从零开始做自动驾驶定位(三): 软件框架

从零开始做自动驾驶定位(四): 前端里程计之初试

从零开始做自动驾驶定位(五): 前端里程计之代码优化

基于数据集实现

4. 里程计精度评价

以组合导航的结果为真值，使用evo工具进行里程计精度评价

1) 安装evo

pip install evo --upgrade --no-binary evo

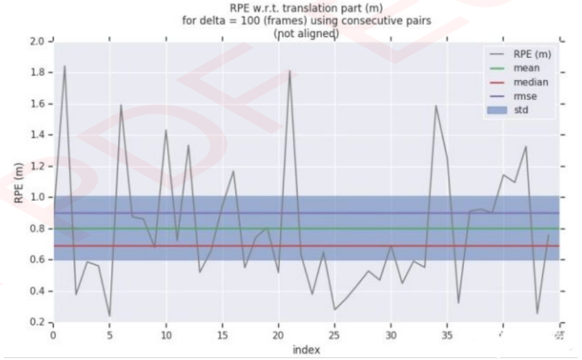
2) 使用evo计算轨迹误差

a. 分段统计精度

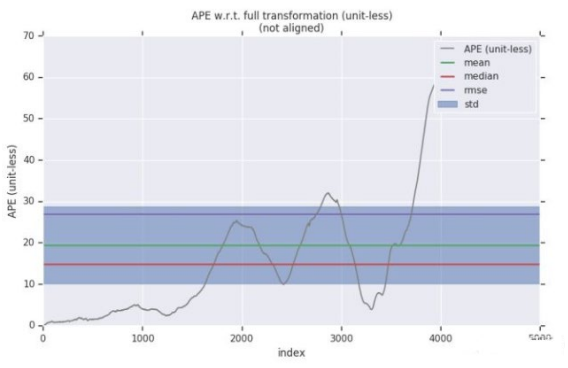
evo_rpe kitti ground_truth.txt laser_odom.txt -r trans_part --delta 100 --plot --plot_mode xyz

b. 计算整体轨迹误差

evo_ape kitti ground_truth.txt laser_odom.txt -r full --plot --plot_mode xyz



分段统计精度



整体轨迹误差

作业

内容：

在提供的工程框架上，结合实际数据集，实现前端激光里程计，并使用evo测试其精度

评价标准：

1) 及格：跑通提供的工程框架

2) 良好：使用evo计算出分段统计误差和整体轨迹误差

3) 优秀：自己实现点云匹配方法，而不是直接调用pcl库的匹配方法，并使用evo计算出指标

ICP实现可参考：https://github.com/tttamaki/SICP-test

深蓝学院
shenlanxueyuan.com

感谢聆听!

Thanks for Listening