基于 LeGO-LOAM 的激光 SLAM 系统设计

学号：122110011209，姓名：彭一凡

摘要：近年来，随着科技发展，各行各业对机器人的需求不断增加。基于 SLAM 的机器人导航定位技术作为实现机器人在未知环境中脱离人为控制，自主移动和作业的基础受到广泛关注。本文基于 LeGO-LOAM 算法框架，完成了激光 SLAM 系统设计，具体工作如下：首先，依据激光雷达种类的不同，对五种主流激光 SLAM 算法进行论述和对比，确定了本文基于 LeGO-LOAM 框架进行系统设计与实现。其次，介绍了 SLAM 的基础理论，明确了 SLAM 要解决的问题，并分析说明了基于列文伯格-马夸尔特优化法（LM）的 SLAM 求解过程。然后，详细分析了 LeGO-LOAM 算法原理。针对 KITTI 数据集的特性，对 LeGO-LOAM 框架进行修改，添加了位姿输出保存功能。最后，基于 KITTI 数据集的 07 数据段对所设计的激光 SLAM 系统进行测试，验证了系统在大规模室外场景下切实有效，能够实现较高精度的定位并建立周围环境的地图。

关键词：LeGO-LOAM，激光 SLAM，定位，地图

一. 引言

1. 研究背景及意义

当今社会，随着移动机器人的大量普及，实现机器人脱离人为控制并在各类场景下自主作业已经成为机器人领域的研究热点。而要解决这一问题，首先要使机器人能够在未知环境中自主移动。因此，机器人导航定位技术逐渐发展起来。Leonard 和 Durrant-Whyte 等人在 1992 年形象地阐述了导航定位所要解决的问题，即“在哪里”，“去哪里”和“怎么去”[1]。目前常见的导航定位方法有 GNSS 导航定位、超声导航定位、惯性导航定位以及 SLAM 导航定位。

SLAM（Simultaneous Localization and Mapping）即同时定位与建图技术，用于解决移动机器人（UAV、UGV 等）在未知环境中实时估计自身位置并在线建立和更新周围环境地图的问题。与基于 GNSS、超声和惯性的方法相比，基于 SLAM 的导航定位技术具有定位精度高、对复杂多样的环境适应性好、能够长时间稳定工作的优点。此外，基于 SLAM 实现导航只需对得到的环境地图进行路径规划，大大降低了导航的难度，减少了系统的计算量。因此，SLAM 成为了目

前最主流的移动机器人导航定位技术，得到了广泛应用。

2. 激光 SLAM 技术研究现状与对比分析

SLAM 根据采集数据所使用的传感器的不同可初步分为基于激光雷达的激光 SLAM 和基于相机的视觉 SLAM。对于激光 SLAM，又可以由激光雷达的线数不同分为基于单线雷达的 2D 激光 SLAM 和基于多线雷达的 3D 激光 SLAM。

(a) 单线激光雷达　　(b) 多线激光雷达

图 1.1 不同类型激光雷达

但随着应用场景的复杂化以及对定位和建图精度要求的不断提高，基于相机、雷达、IMU、轮式编码器、GPS 等多种传感器融合的 SLAM 系统已经越来越成为主流的 SLAM 方案。

下面，本文将对五种主流的激光 SLAM 方法进行论述和分析。

(1) Gmapping

Gmapping 是一种基于 RBPF 粒子滤波的 2D 激光 SLAM 方法。RBPF 粒子滤波算法，即将定位和建图过程分离，先进行定位再进行建图。这种方法融合了里程计和激光信息，使得每个粒子都携带一个地图。该算法构建小场景地图所需的计算量较小，精度较高。但在高分辨率建图时，在静止状态下更新效果较差，存在震荡且噪声过多。同时，由于粒子滤波方法只估计当前的状态，不修正以前的状态，因此地图越大，误差累计越多，导致 Gmapping 不适用于大规模场景建图[2]。

(2) Cartographer

Cartographer 是由谷歌开源的一种基于图优化的 2D 激光 SLAM 算法。该方法由前端激光激光里程计、后端优化和回环检测三部分组成。Cartographer 模块化、系统化、工程化程度很高，封装完善，同时具备传感器同步、位姿外推器、激光数据预处理（去畸变、重复点云删除）功能。前端在帧匹配环节融合了暴力

匹配法和基于 LM 的梯度优化方法，后端优化过程基于 SPA 完成，同时还引入了 submap 的概念，以便于回环检测上使用分支定界的方法进行快速搜索。不仅如此，Cartographe 还开放了 landmark、GPS 等数据融合的接口，提供了地图续扫和定位的功能[3]。

(3) LOAM

LOAM（Lidar Odometry and Mapping）是一种经典的 3D 激光 SLAM 框架。其主要思想是通过一个高频激光里程计进行低精度的运动估计，即使用激光雷达做里程计，计算两次扫描之间的位姿变换。同时，LOAM 还包含另一个执行低频但是高精度的建图与校正里程计，利用多次扫描的结果构建地图，细化位姿轨迹。在点云匹配与特征提取时，由于 scan-to-scan 匹配精度低但速度快， map-to-map 匹配精度高但是速度慢，创新性使用 scan-to-map 来兼具精度与速度，这种思路给后续很多基于激光里程计或多传感器融合框架提供思路。但是原本的 LOAM 没有 IMU 辅助，不带有回环检测，因而不可避免引起漂移误差[4]。

(4) LeGO-LOAM

LeGO-LOAM 是一种基于地面优化的轻量级 SLAM 系统。其在 LOAM 的基础上细化了特征提取与优化，且带有回环功能。LeGO-LOAM 的主要流程与 LOAM 大致相同，区别在于 LeGO-LOAM 在 LOAM 基础上采用地面分割方式将点云分为地面点与非地面点，进一步缩小特征提取范围，以加快计算速度，能够在嵌入式设备上实时运行；在建图模块添加了位姿图与回环检测，解决了 LOAM 没有后端优化的弊端，提高了建图精度和效率[5]。

(5) LIO-SAM

LIO-SAM 是一种基于平滑和匹配的紧耦合 3D 激光惯性 SLAM 框架。LIO-SAM 在因子图（factor graph）上方制定了激光雷达惯性里程表，从而可以将不同来源的大量相对和绝对测量值作为因子合并到系统中，通过 IMU 预积分的估计运动使点云偏斜，并为激光雷达里程计优化提供了初始预测。为了确保实时高性能，LIO-SAM 将之前的激光雷达扫描边缘化以进行位姿优化，而不是将激光雷达扫描与全局地图匹配。该算法选择性地引入关键帧，同时以有效的滑动窗口将新的关键帧注册到固定大小的先验“子关键帧”集合中，在局部范围而不是全局范围内进行扫描匹配[6]。

目前，诸如扫地机器人一类的低速地面移动机器人多基于 2D 激光 SLAM 如 Gmapping 和 Cartographer 实现定位和建图功能，而对于物流车、无人机和自动驾驶汽车等则多搭载多线激光雷达以实现高精度的定位和获得周围环境的稠密点云地图以用于导航或语义理解。

目前，工程中应用较多的激光 SLAM 算法是 LOAM 框架。但是 LOAM 存在运算量大，迭代次数多的问题。LeGO-LOAM 算法在 LOAM 算法基础上进行改进。通过点云分割聚类、点云降采样、特征点筛选与空间分配等方法，减少了激光里程计中匹配过程的迭代计算量。通过对点云与地图特征点的匹配得到更为准确的位姿估计结果。根据该结果校正点云，完成点云地图的构建与更新。因此，本文将基于 LeGO-LOAM 框架进行激光 SLAM 系统设计。

3.本文主要内容

本文完成了基于 LeGO-LOAM 的激光 SLAM 系统设计。首先建立和求解了 SLAM 数学模型。然后对 LeGO-LOAM 各部分实现原理进行了详细分析，并采用 KITTI 数据集对所设计的激光 SLAM 系统进行验证。针对 KITTI 数据集的特性，修改 LeGO-LOAM 的参数，并为其添加输出保存 KITTI 格式的全局位姿的功能，以便于和 KITTI 官方发布的位姿真值进行对比，验证系统有效性和精度。

二. 激光 SLAM 理论基础

1.SLAM 数学模型

SLAM 要解决定位与建图两大问题。对于定位问题，即必须确定机器人身处何地，对应着 SLAM 运动模型；而对于建图问题，则是确定机器人周围的环境是什么样的，对应着 SLAM 观测模型。因此，SLAM 数学模型包含运动模型和观测模型[7]。

这里首先引入“位姿”这一概念。位姿可以理解为位置和姿态，对应机器人空间坐标和角度变换，定位问题本质上是利用传感器数据对机器人位姿进行解算。将机器人运动过程离散化，对应时刻 1~k，每一个时刻对应的位姿分别为$x_1 \cdots x_k$。机器人的运动方程可以利用抽象函数f来表示：

$$x_k = f(x_{k-1}, u_k) + w_k \tag{2.1}$$

式(2.1)中，x_{k-1}为 k-1 时刻的位姿，u_k为传感器输入数据，w_k表示传感器采集数据的噪声，f是一个通用表达式，并未赋予其特定的表达式，这也就意味着

不同机器人在不同环境下运动得到的f函数是不相同的。

对于建图问题，设机器人在 1~k 时刻共观测到 N 个路标点$y_1 \cdots y_N$，机器人位姿为x_k时观测到路标点y_j，这样就产生了一个用于描述观测情况的观测向量$z_{j,k}$。同样地，用抽象函数 g 来描述机器人观测方程：

$$z_{j,k} = g(x_k, y_j) + v_{j,k} \tag{2.2}$$

式(2.2)中，$v_{j,k}$表示观测噪声。

基于上述分析，SLAM 的数学模型可以进一步表述为利用传感器的输入u_k以及 k 时刻的观测情况$z_{j,k}$来求解机器人具体位置x_k和观测点y_j的条件概率分布问题。根据贝叶斯法则，可以将 SLAM 问题描述如下：

$$P(x_k, y_j|z_{j,k}, u_k) = \frac{P(z_{j,k}, u_k|x_k, y_j)P(x_k, y_j)}{P(z_{j,k}, u_k)} \propto P(z_{j,k}, u_k|x_k, y_j)P(x_k, y_j) \tag{2.3}$$

对于式(2.3)，$P(x_k, y_j|z_{j,k}, u_k)$又称为后验概率，$P(z_{j,k}, u_k|x_k, y_j)$称为似然，$P(x_k, y_j)$称为先验。由于机器人所处环境未知，因此机器人位置和环境的先验信息是无法有效获取的。此时，若要求解后验概率的最优估计，可以将该状态下的后验概率最大化，即将该问题退化为最大似然估计（Maximize Likelihood Estimation, MLE）问题进行求解。

$$MLE = \arg\max P(z_{j,k}, u_k|x_k, y_j) \tag{2.4}$$

对于噪声$v_{j,k}$和w_k，通常假设$v_{j,k}$其符合均值为 0，方差为$Q_{j,k}$的高斯分布，w_k符合均值为 0，方差为R_k的高斯分布，即:

$$\begin{aligned} v_{j,k} &\sim \mathcal{N}(0, Q_{j,k}) \\ w_k &\sim \mathcal{N}(0, R_k) \end{aligned} \tag{2.5}$$

故无输入u_k的观测方程概率分布可以表示为：

$$P(z_{j,k}|x_k, y_j) = N(g(x_k, y_j), Q_{j,k}) \tag{2.6}$$

基于最小化负对数法求解上式得：

$$-\ln\left(P(z_{j,k}|x_k, y_j)\right) = \frac{1}{2}\ln\left((2\pi)^N \det(Q_{j,k})\right) + \tag{2.7}$$

$$\frac{1}{2}(z_{j,k}-g(x_k,y_j))^T Q_{j,k}^{-1}(z_{j,k}-g(x_k,y_j))$$

由于对数函数是单调的，因此对原函数求最大化相当于对负对数求最小化。对于式(2.7)，由于第一项与$z_{j,k}$无关，因此只要最小化右边的二次型就可以得到对状态的最大似然估计[7]。

$$\arg\max P(z_{j,k}|x_k,y_j)=\arg\min(z_{j,k}-g(x_k,y_j))^T Q_{j,k}^{-1}(z_{j,k}-g(x_k,y_j)) \quad (2.8)$$

通常地，认为各个时刻的输入和观测是相互独立的，因此有：

$$P(z_{j,k},u_k|x_k,y_j)=\prod_k P(u_k|x_{k-1},x_k)\prod_{k,j}P(z_{j,k}|x_k,y_j) \quad (2.9)$$

定义误差如下：

$$\begin{aligned} e_{\boldsymbol{u},k}&=x_k-f(x_{k-1},u_k)\\ e_{\boldsymbol{z},j,k}&=z_{j,k}-g(x_k,y_j)\end{aligned} \quad (2.10)$$

最终求解方程如下：

$$J(x_k,y_j)=\sum_j(e_{\boldsymbol{u},k})^T R_k^{-1}(e_{\boldsymbol{u},k})+\sum_k\sum_j(e_{\boldsymbol{z},j,k})^T Q_{j,k}^{-1}(e_{\boldsymbol{z},j,k}) \quad (2.11)$$

2.SLAM 问题求解

基于上述分析，求解 SLAM 问题就是在求解$J(x_k,y_j)$的极小值。基于优化的方法构建最小二乘问题：

$$\min J(x_k,y_j)=\sum_j(e_{\boldsymbol{u},k})^T R_k^{-1}(e_{\boldsymbol{u},k})+\sum_k\sum_j(e_{\boldsymbol{z},j,k})^T Q_{j,k}^{-1}(e_{\boldsymbol{z},j,k}) \quad (2.12)$$

下面基于列文伯格-马夸尔特方法（LM 法）对上式进行求解，为方便表述，规定机器人状态$\boldsymbol{x}=(x_1\cdots x_k,y_1\cdots y_j)$。

LM 法实际上是对高斯牛顿法的一种改进方法，本文在此不对高斯牛顿法的详细步骤进行赘述，仅对 LM 法求解 SLAM 问题的原理进行分析。高斯牛顿法的主要思想是对$J(x)$在x_0点进行一阶泰勒展开，并对一阶展开式的二范数进行极值求解。但是该方法只在展开点附近有较好的效果，当$\Delta\boldsymbol{x}$超出一定范围以后，拟合和近似效果会明显下降。针对此问题，LM 法引入信赖区域，即给增量Δx添加

一个范围，将无约束问题转化为有约束问题进行求解[7]。

定义近似效果评价因子ρ：

$$\rho = \frac{J(\boldsymbol{x_0} + \Delta \boldsymbol{x}) - J(\boldsymbol{x_0})}{\boldsymbol{\varphi}(\boldsymbol{x_0})^{\mathrm{T}} \Delta \boldsymbol{x}} \tag{2.13}$$

其中，$\boldsymbol{\varphi}(\boldsymbol{x_0})^{\mathrm{T}}$表示 $J(\boldsymbol{x})$在$\boldsymbol{x_0}$点的雅克比矩阵，即导数。

定义系数矩阵$\boldsymbol{D}$，通常规定$\boldsymbol{D}$为非负对角矩阵，其取值为$\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{\varphi}$的对角元素平方根。给定初始优化半径$\mu$，构建约束条件如下：

$$||\boldsymbol{D}\Delta \boldsymbol{x}_k||^2 \leq \mu \tag{2.14}$$

构建有约束的最小二乘问题如下：

$$\arg\min ||\boldsymbol{J}(\boldsymbol{x_0} + \Delta \boldsymbol{x})||, \ ||\boldsymbol{D}\Delta \boldsymbol{x}_k||^2 \leq \mu \tag{2.15}$$

使用拉格朗日乘子将式(2.15)转换为无约束非线性最小二乘方程如下：

$$\min \frac{1}{2}||\boldsymbol{J}(\boldsymbol{x}_k) + \boldsymbol{\varphi}(\boldsymbol{x}_k)\Delta \boldsymbol{x}_k||^2 + \frac{\lambda}{2}(\left||\boldsymbol{D}\Delta \boldsymbol{x}_k\right|\!|^2 - \mu) \tag{2.16}$$

对式(2.16)求导，令其关于 $\boldsymbol{x}$ 导数为零，可得：

$$(\boldsymbol{\varphi}(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{\varphi}(\boldsymbol{x}_k) + \lambda_k \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D})\Delta \boldsymbol{x}_k = -\boldsymbol{\varphi}(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x}_k) \tag{2.17}$$

通过解式(2.17)可以得到增量$\Delta \boldsymbol{x}_k$。

基于上述分析，可以将 LM 法的具体流程描述如下：

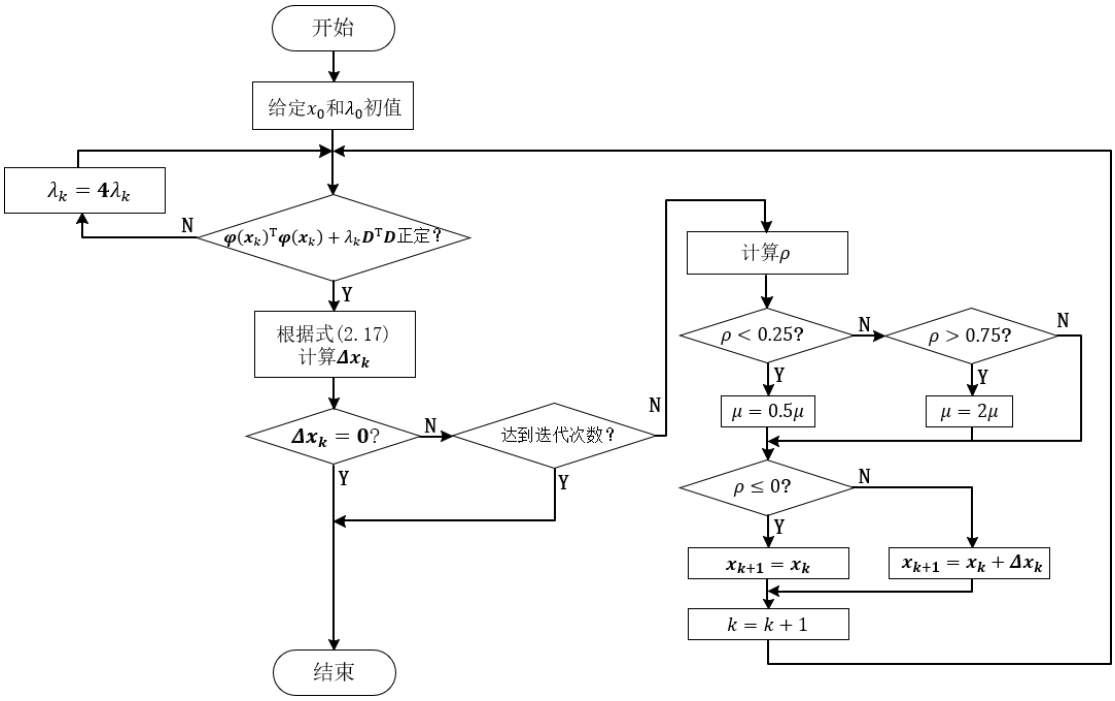


图 2.1 LM 法流程图

以上完成了 LM 优化法的理论分析，这部分内容作为后文中 LeGO-LOAM 算法点云优化与位姿解算的基础将被反复应用到，具有重要意义。

三. 基于 LeGO-LOAM 的激光 SLAM 系统设计

1. LeGO-LOAM 算法流程

LeGO-LOAM 与 LOAM 相同，将位姿估计问题划分为高频低精度位姿估计和低频高精度位姿估计两个不同的部分，最后将两种估计结果融合，实现高频高精度的位姿估计。此外，LeGO-LOAM 在分割和优化的过程中利用了地面的约束，能够在减少计算量的情况下提升定位和建图的精度[5]。

LeGO-LOAM 的输入为激光雷达点云数据，输出为估计位姿，算法框架如下：

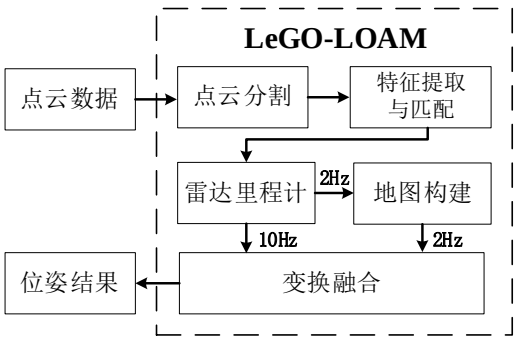


图 3.1 LeGO-LOAM 系统框架

由图 3.1 可知，LeGO-LOAM 系统由五部分组成。点云分割负责分割分离地面点，并对点云进行聚类处理。特征提取与匹配部分基于点云分割的结果提取点云特征，并进行帧间特征匹配。激光里程计基于特征点在相邻帧之间的位置变化以 10Hz 的频率进行位姿估计。建图模块对激光里程计输出的位姿信息进一步处理，同时将去畸变点云注册到全局地图中，并以 2Hz 的频率更新地图和位姿。变换融合模块接收来自激光里程计模块和雷达建图模块的高低频位姿估计结果，并以 10Hz 的频率对其进行融合输出。

2. 点云分割与聚类

在进行点云分割与聚类之前需要先对点云进行重投影操作，将每一帧点云的所有点投影到一张深度图像（range image）上，并为每一个点计算其行号（rowIdn）和列号(columnIdn)。对于一个 16 线激光雷达来说，若其水平分辨率(ang_res_x)为0.2°，垂直分辨率(ang_res_y)为2°，则每扫描一圈会得到一个16×1800的点云阵列，垂直角度θ_v范围为$[-15°, 15°]$，水平角度θ_h范围为$[0°, 360°]$。

行号计算方法如下：

$$\text{rowIdn} = \frac{\theta_v + 15}{2} \tag{3.1}$$

在雷达坐标系下，每一个点的空间坐标如下图 3.2 所示：

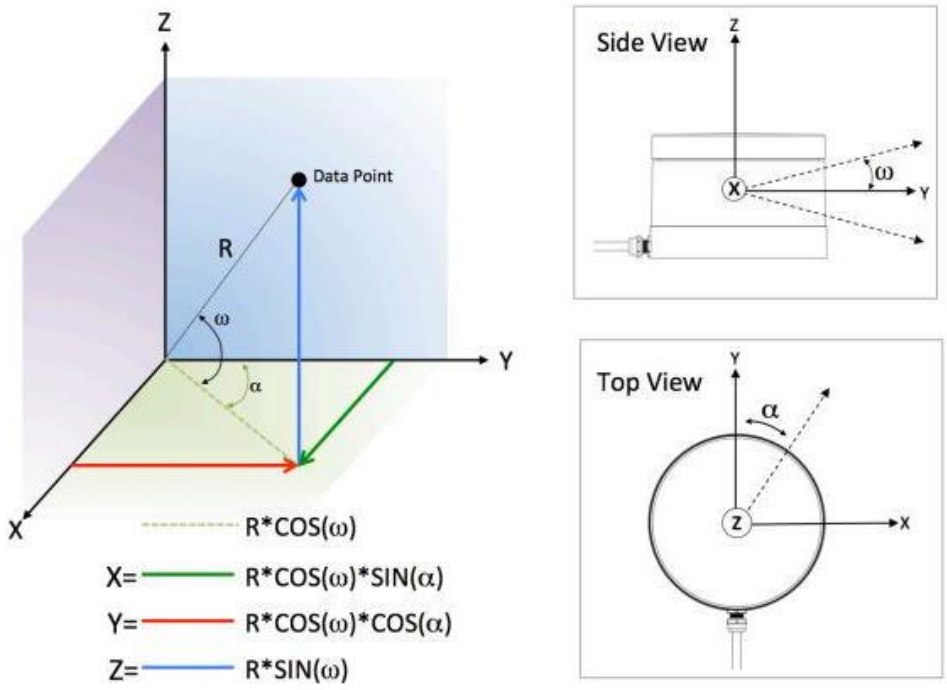


图 3.2 点云空间坐标示意图

水平夹角$\alpha = \arctan\frac{x}{y} \times \frac{180}{\pi}$，故有列号计算方法如下：

$$\text{columnIdn} = \frac{\alpha - 90}{\text{ang\_res\_x}} + \frac{\text{Horizon\_SCAN}}{2} \tag{3.2}$$

这里要特别说明的是，式(3.2)中的Horizon_SCAN代表一束激光扫描一圈得到的点数，在系统中还会定义 N_SCAN 参数，代表雷达线数。定义这些参数的意义在于使用不同线数不同垂直或水平分辨率的雷达时，方便对程序进行修改。

行列号计算完毕后，再对深度图 range image 中的每一个点赋予传感器获得的强度信息，即距离，完成将点云投影至深度图上的操作。

（1）点云分割

这里首先对点云这一概念进行介绍。激光雷达在扫描的过程中会获得周围环境的点云信息，每一个点云都包含 XYZ 坐标以及强度（Intensity）信息。XYZ 坐标是点云在传感器坐标下的位置，强度信息是指探测点处物体的反射强度，与深度（距离）相关[8]。

LeGO-LOAM 首先对原始点云数据进行点云分割，其目的是为了分离地面点

云，减少后续特征匹配和位姿解算的运算量。对于水平安装的 16 线激光雷达来说，由于激光的序列号 0~15 是从下向上排列的，因此地面只可能在前七束激光中提取。故只需对前七束激光获得的点云数据进行遍历判断即可。

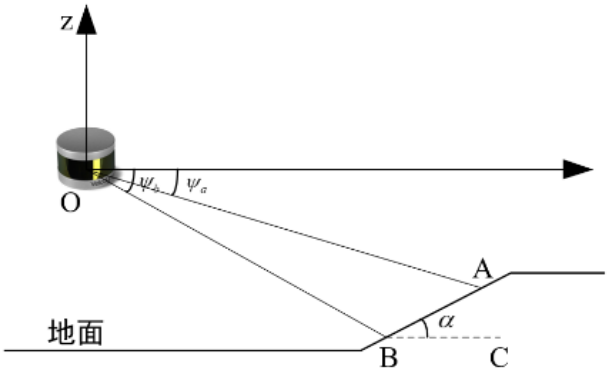


图 3.3 基于角度的地面分割示意图

点云分割的主要思想是利用角度对地面进行判断，进而分离地面点。如图 3.2 所示，设雷达第 i 束激光扫描得到 A 点，第 i+1 束激光对应扫描得到 B 点，则利用点云信息中包含的 XYZ 坐标参数可以计算两点之间的坐标差向量$\overrightarrow{AB} = (dx, dy, dz)$。则若该差向量$\overrightarrow{AB}$与水平面的夹角$\alpha$小于一个阈值角度即可以判定 A、B 为地面点。本文设定阈值角度为10°，角度计算公式如下：

$$\alpha = \arctan\frac{dz}{\sqrt{(dx)^2 + (dy)^2}} \times \frac{180}{\pi} \tag{3.3}$$

遍历全部点云后，对于地面点进行标记，不对其作点云聚类操作。

（2）点云聚类

本文基于广度优先搜索（BFS）实现点云聚类，其主要思想是将对距离的判断转化为对角度的判断，即用两点连线的夹角来判断这两点是否属于同一物体。

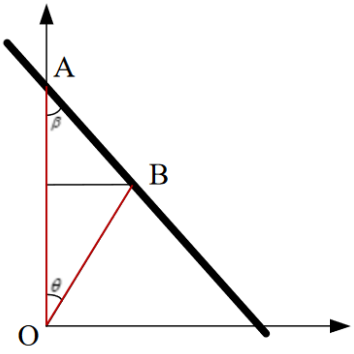


图 3.4 点云聚类示意图

如图 3.3 所示，OA 和 OB 是两条激光束，其距离值可以由激光雷达获得。以 OA 为距离较长的一束为例，定义 OA 与 AB 之间的夹角为β，则β的计算方法可分为水平方向和垂直方向两类。

若 A 和 B 是同一水平面上的两个点，则有：

$$\beta = \arctan\frac{|OB| * \sin(ang\_res\_x)}{|OA| - |OB| * \cos(ang\_res\_x)} \times \frac{180}{\pi} \tag{3.4}$$

若 A 和 B 是同一垂直面上的两个点，则有：

$$\beta = \arctan\frac{|OB| * \sin(ang\_res\_y)}{|OA| - |OB| * \cos(ang\_res\_y)} \times \frac{180}{\pi} \tag{3.5}$$

通常认为当$\beta > 60°$时，A 和 B 两点属于同一物体。

BFS 的主要思想是，对于每一帧点云的深度图，从左上角作为起始点，依次搜索它的上、下、左、右四个非地面点。如果判定它们和起始点同属于一个物体，就把它们放入队列，并赋予相同的标记值；从队列中删去该起始点，重新选取队列中的第一个点为起始点，仍然遍历该起始点的上下左右点循环往复直到队列为空。至此认为聚类出一个完整的物体，并对其余未标记点重复上述遍历过程[9,10]。

3. 点云特征提取与匹配

（1）特征提取

点云特征提取是指对分割聚类后的点云提取特征，以便于后续的特征匹配和位姿解算。LeGO-LOAM 将特征点分为两大类，一类是平面特征点，又称平面特征，另一类是边缘特征点，又称角点。两类特征点可以通过计算曲率值来进行区分[11]。

(a) 平面特征　　(b) 边缘特征

图 3.5 两类点云特征

设 S 为深度图中某一点p_i同一行的连续点集，且 S 中各有一半的点在p_i两

侧。本文规定$|S|=10$，故p_i两侧各有五个点。曲率 c 计算公式如下：

$$c=\frac{1}{|S|\cdot\left\|X_i^L\right\|}\left\|\sum_{j\in S,j\neq i}(X_i^L-X_j^L)\right\| \tag{3.6}$$

其中，X_i^L和X_j^L分别代表激光雷达坐标系下t_k时刻 i 和 j 两点的空间坐标。当 c 较大时为边缘特征点，c 较小时为平面特征点

为了避免选取的特征过于集中在同一位置，本文将激光雷达的扫描点按角度划分为 6 个子区域，将特征点按曲率值大小分为角点、次角点、次平面点和平面点四类。对每一份子区域中所有点的曲率进行排序，选取曲率值最大的 2 个非地面点作为角点，20 个曲率次大的点作为次角点，曲率最小的 4 个地面点作为平面点，而地面或分割点云中不属于角点或次角点的作为次平面点。

（2）特征匹配

特征匹配是指用相邻两帧点云中提取出的线特征和面特征点进行点云配准，是后面进行位姿解算的基础。设t_k时刻获得去畸变的点云P_k^+，t_{k+1}时刻获得去畸变的点云P_{k+1}^-，则根据特征点提取结果，可以提取上一帧点云P_k^+和当前帧点云P_{k+1}^-的角点和平面点。对于P_{k+1}^-中的特征点，需要在P_k^+中寻找与之最近的点构造对应关系。

由位姿变换理论可知，空间中的两点之间的位姿变换可以由一次旋转和一次平移来描述。设当前帧中有一点 i，坐标表示为$\tilde{X}_{(k+1,i)}^L$，经过一次旋转 **R** 和一次平移$\boldsymbol{T}_{(k+1,i)}^L$得到一个新的坐标点$X_{(k+1,i)}^L$，即：

$$X_{(k+1,i)}^L=\boldsymbol{R}\tilde{X}_{(k+1,i)}^L+\boldsymbol{T}_{(k+1,i)}^L \tag{3.7}$$

当前帧和前一帧中的点云坐标可以通过上式互相转换。特征匹配的主要思想是用空间中的欧式距离来评价匹配效果的好坏，即将当前帧中的一个点通过式(3.7)变换到前一帧中，并找到前一帧的点云中与该变换点欧式距离最小的点，作为一对匹配点。此时的旋转 **R** 和平移$\boldsymbol{T}_{(k+1,i)}^L$就是匹配点之间的最优坐标转换。这里要特别说明的是，LeGO-LOAM 算法采用点与线和点与面之间的距离代替点与点之间的距离进行匹配判断，以求得更为准确的匹配结果。

角点特征匹配过程如图 3.6 所示。设 i 是P_{k+1}^-中的一个角点，P_k^+中需要两

个点确定与 i 对应的边缘线。对于 i，设 j 是P_k^+中与 i 最近的点，l 是P_k^+中与 j 所在扫描面相邻的两个扫描面所得点云中与 i 最近的角点。则由 j 和 l 确定一条直线，只需不断计算 i 与该直线的距离直到收敛就可以判断得到与 i 对应的边缘线。这一距离 d 可由下式计算得到：

$$d = \frac{\left|(\bar{X}^L_{(k+1,i)} - \bar{X}^L_{(k,j)}) \times (\bar{X}^L_{(k+1,i)} - \bar{X}^L_{(k,l)})\right|}{|\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,l)}|} \tag{3.8}$$

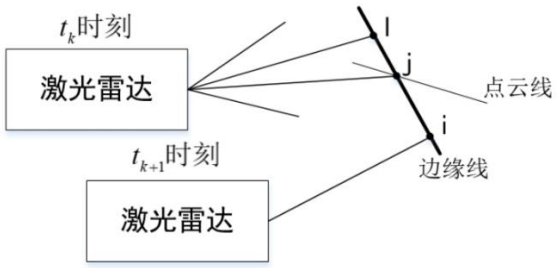


图 3.6 相邻帧角点特征匹配示意图

平面特征匹配过程如图 3.7 所示。设 i 是P_{k+1}^-中的一个平面点，P_k^+中需要三个点确定与 i 对应的平面。对于 i，设 j 是P_k^+中与 i 最近的点，l 和 k 是P_k^+中与 j 所在扫描面相邻的两个扫描面所得点云中与 i 最近的平面特征点。则由 j 和 l 确定一条直线，k 在该直线外与之构成一个平面。只需不断计算 i 与该平面的距离直到收敛就可以判断得到与 i 对应的平面。这一距离 d 可由下式计算得到：

$$d = \frac{\left|(\bar{X}^L_{(k+1,i)} - \bar{X}^L_{(k,j)}) \cdot (\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,l)}) \times (\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,k)})\right|}{|(\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,l)}) \times (\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,k)})|} \tag{3.9}$$

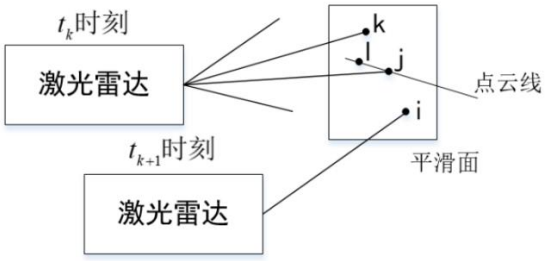


图 3.7 相邻帧平面特征匹配示意图

4.Lidar 里程计

实际上，无论是角点特征匹配还是平面特征匹配，距离 d 都可以视为观测误差，这样一来就把位姿解算问题转换为了 SLAM 问题的本质，即求解最小二乘问题以最小化误差 d，即得到最优的位姿变换。这里的最小二乘问题可以基于前

文所述的 LM 法进行求解。

对于角点特征匹配，进行位姿解算可以计算出 6 自由度位姿中的$[t_x, t_y, \theta_{yaw}]$，即水平方向的位移和航向角的帧间变化。对于平面点特征匹配，则可以计算得到$[t_z, \theta_{roll}, \theta_{pitch}]$，即俯仰角、滚转角和高度的帧间变化。因此，本文要使用两次 LM 优化法来解算出 6 自由度位姿。

将 d 表示为关于$X_{(k+1,i)}^{L}$的函数：

$$d = f\left(X_{(k+1,i)}^{L}\right) = f(\boldsymbol{R}\tilde{X}_{(k+1,i)}^{L} + \boldsymbol{T}_{(k+1,i)}^{L}) \tag{3.10}$$

式(3.10)对θ_{yaw}求偏导得：

$$\begin{aligned}\frac{\partial d}{\partial \theta_{yaw}} &= \frac{\partial f(\boldsymbol{R}\tilde{X}_{(k+1,i)}^{L} + \boldsymbol{T}_{(k+1,i)}^{L})}{\partial \theta_{yaw}} \\ &= \frac{\partial f(\boldsymbol{R}\tilde{X}_{(k+1,i)}^{L} + \boldsymbol{T}_{(k+1,i)}^{L})}{\partial X_{(k+1,i)}^{L}} \cdot \frac{\partial(\boldsymbol{R}\tilde{X}_{(k+1,i)}^{L} + \boldsymbol{T}_{(k+1,i)}^{L})}{\partial \theta_{yaw}} \\ &= \frac{\partial f\left(X_{(k+1,i)}^{L}\right)}{\partial X_{(k+1,i)}^{L}} \cdot \frac{\partial \boldsymbol{R}\tilde{X}_{(k+1,i)}^{L}}{\partial \theta_{yaw}}\end{aligned} \tag{3.11}$$

同理还可以求得 d 关于$\theta_{roll}, \theta_{pitch}$的偏导。

式(3.10)对t_x求偏导得：

$$\begin{aligned}\frac{\partial d}{\partial t_x} &= \frac{\partial f(\boldsymbol{R}\tilde{X}_{(k+1,i)}^{L} + \boldsymbol{T}_{(k+1,i)}^{L})}{\partial t_x} \\ &= \frac{\partial f(\boldsymbol{R}\tilde{X}_{(k+1,i)}^{L} + \boldsymbol{T}_{(k+1,i)}^{L})}{\partial X_{(k+1,i)}^{L}} \cdot \frac{\partial(\boldsymbol{R}\tilde{X}_{(k+1,i)}^{L} + \boldsymbol{T}_{(k+1,i)}^{L})}{\partial t_x} \\ &= \frac{\partial f\left(X_{(k+1,i)}^{L}\right)}{\partial X_{(k+1,i)}^{L}}\end{aligned} \tag{3.12}$$

同理还可以求得 d 关于t_y, t_z的偏导。

这样以来便可以得到 d 的雅克比矩阵 $\boldsymbol{J}$，进一步构建最小二乘问题，并用 LM 法迭代求解，使得 d 趋向于 0。

基于上述分析，可以将激光里程计解算位姿的流程描述如下：

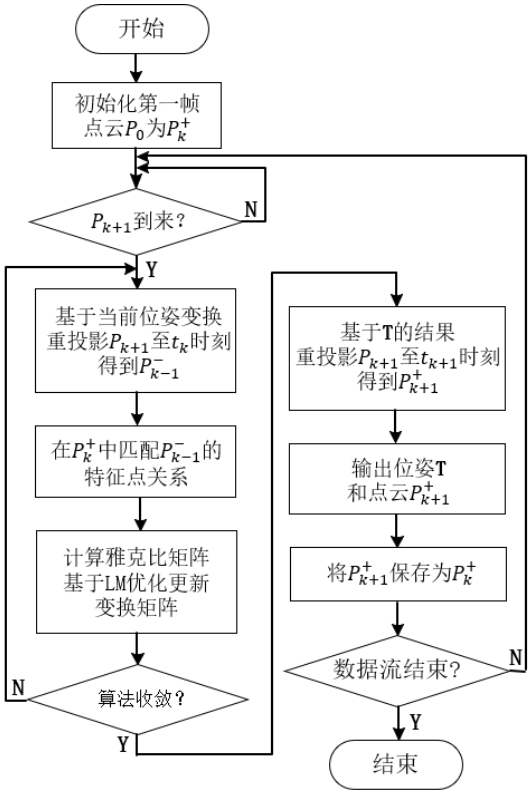


图 3.8 激光里程计解算位姿流程图

5.位姿优化与地图构建

在上一部分中已经完成了对某一时刻位姿的解算和优化。但是，由于这种局部的解算和优化只针对当前帧并且不能完全消除位姿误差，导致在机器人运行的过程中位姿误差会不断的积累。为解决这一问题，必须要对全局的位姿进行优化。

完成对t_{k+1}时刻点云P_{k+1}^-的估计后，激光里程计输出一个重投影后的点云P_{k+1}^+以及t_k到t_{k+1}时间段内激光雷达运动的姿态变换T_k^L。建图算法在地图坐标系中匹配、处理当前帧点云P_{k+1}^+并更新地图点云。定义最近一次更新的地图点云即t_n时刻更新的地图点云为Q_n。建图算法在t_n时刻更新的激光雷达在地图中的位姿估计定义为T_n^M。根据上一次更新的雷达在地图中的位姿T_n^M与t_n时刻开始到当前时刻t_{n+1}为止时间段内激光里程计的运动估计的累计量 T_{n+1}^L，计算得到当前时刻雷达位姿的初值T_{n+1}^M。根据T_{n+1}^M，将最新的点云注册到地图坐标系中，得到投影后的点云P_{n+1}^M。接下来，算法匹配P_{n+1}^M与Q_n中的特征，并最小化特征距离优化求解T_{n+1}^M[8,11,12]。

激光里程计以 10Hz 频率运行，为地图构建与优化模块提供先验信息。而建图模块以 2Hz 的频率运行，解算更精确的位姿以及更新地图。这样既可以实现高精度的位姿估计和地图构建，又保证了位姿输出的频率。

6.位姿输出保存

原始的 LeGO-LOAM 框架是没有位姿输出和保存功能的，本文为了更加直观的评估位姿估计的结果，为其添加在文本文件中保存全部估计位姿的功能。这里要特别说明的是，由于本文选定 KITTI 数据集对激光 SLAM 系统进行测试，因此要输出准确的估计位姿，必须要首先针对 KITTI 数据集的特性修改代码参数。

KITTI 数据集是基于 HDL-64 线激光雷达录制,,且其一次扫描的点数为 2083。相应地，其水平和垂直分辨率也需要进行修改。最后，由于 HDL-64 的激光仰角比较小，因此本文将点分割环节中扫描地面需要的激光线数增加至 55。具体修改如下：

```
extern const int N_SCAN = 64;
extern const int Horizon_SCAN = 2083;
extern const float ang_res_x = 360.0/float(Horizon_SCAN);
extern const float ang_res_y = 26.8/float(N_SCAN-1);
extern const float ang_bottom = 24.8;
extern const int groundScanInd = 55;
```

图 3.9 针对 KITTI 数据集修改的部分参数

修改过参数后，可以在 laserOdometryHandler 函数中加入位姿保存的代码，这里需要注意的是，KITTI 数据集位姿真值格式为每个位姿是一个 1 行 12 列的矩阵，前九个值代表了旋转矩阵，后三个值代表位移值。

```
laserOdometry2.header.stamp = laserOdometry->header.stamp;
laserOdometry2.pose.pose.orientation.x = -geoQuat.y;
laserOdometry2.pose.pose.orientation.y = -geoQuat.z;
laserOdometry2.pose.pose.orientation.z = geoQuat.x;
laserOdometry2.pose.pose.orientation.w = geoQuat.w;
laserOdometry2.pose.pose.position.x = transformMapped[3];
laserOdometry2.pose.pose.position.y = transformMapped[4];
laserOdometry2.pose.pose.position.z = transformMapped[5];

Eigen::Quaterniond q;
q.w()=laserOdometry2.pose.pose.orientation.w;
q.x()=laserOdometry2.pose.pose.orientation.x;
q.y()=laserOdometry2.pose.pose.orientation.y;
q.z()=laserOdometry2.pose.pose.orientation.z;
Eigen::Matrix3d R = q.toRotationMatrix();
```

图 3.10 位姿输出与保持核心代码

四. 实验与分析

1. 实验环境

（1）软件系统与依赖库

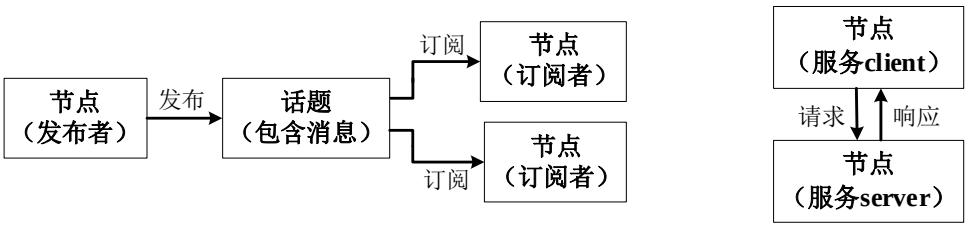
本文所设计的激光 SLAM 系统软件运行环境为 Ubuntu18.04 操作系统，并基于 ROS-melodic 运行。

ROS 的主要组成部分及其作用如表 4.1 所示。

表 4.1 ROS 基本组成

名称	功能
节点（node）	执行单元（相当于进程）每一个节点在系统中完成一个功能。节点在系统中的名称位移
话题（topic）	话题是节点之间通信的“桥梁”，一个节点发布或订阅话题的过程就是在进行发送或接收信息的过程。同一节点往往能够同时发布或订阅多个话题，通信效率很高。
消息(messang)	消息是话题包含的数据，传感器数据、程序运行的结果等都可以视为消息。
服务(service)	服务是另一种通信形式，用于实现节点间同步通信。

节点、消息、话题和服务的关系可以用下图来表示：



(a) 节点间话题通信　　(b) 节点间通信

图 4.1 ROS 通信示意图

本文代码的集成开发环境为 Ubuntu18.04 下的 CLion 编译器。该编译器支持 cmake 工程，并且能够实现一键编译直接生成可执行文件，提高了 cmake 工程的编译效率。

运行代码所需要安装的各类库及其对应版本如下表所示：

表 4.2 依赖库及其对应版本

名称	版本
OpenCV	3.4.3
Gtsam	4.0.0-alpha2
Python	2.7.17
ceres	2.0.0
PCL	1.8.1
Eigen	3.3.4
CMake	3.10.2

对于表 4.2 中的各类库的安装方法，本文在此不多加赘述。

（2）KITTI 数据集

本文选用 KITTI 数据集对系统进行测试。KITTI 数据集由德国卡尔斯鲁厄理工学院和丰田美国技术研究院联合创办，是目前国际上最大的自动驾驶场景下的计算机视觉算法评测数据集。该数据集用于评测立体图像(stereo)，光流(optical flow)，视觉测距(visual odometry)，3D 物体检测(object detection)和 3D 跟踪(tracking)等计算机视觉技术在车载环境下的性能。

KITTI 包含市区、乡村和高速公路等场景采集的真实图像数据，分为不同的数据段供研究人员下载。最重要的是，KITTI 数据集提供了 00-10 共 11 个数据段的位姿真值，便于评估不同 SLAM 系统的效果。

(a) KITTI 数据部分数据段

```
00: 2011_10_03_drive_0027
01: 2011_10_03_drive_0042
02: 2011_10_03_drive_0034
03: 2011_09_26_drive_0067
04: 2011_09_30_drive_0016
05: 2011_09_30_drive_0018
06: 2011_09_30_drive_0020
07: 2011_09_30_drive_0027
08: 2011_09_30_drive_0028
09: 2011_09_30_drive_0033
10: 2011_09_30_drive_0034
```

(b) 真值文件及其对应的数据段

图 4.2 KITTI 数据集

（3）可视化

本文基于 ROS 系统下的 rviz 软件对定位与建图过程进行可视化。在 rviz 中可以实时查看基于 KITTI 数据集运行本文系统所得到的轨迹、分割点云以及稠

密点云地图等信息。除此以外，rviz 还可以清晰表达 ROS 中各个部分传感器的坐标关系。

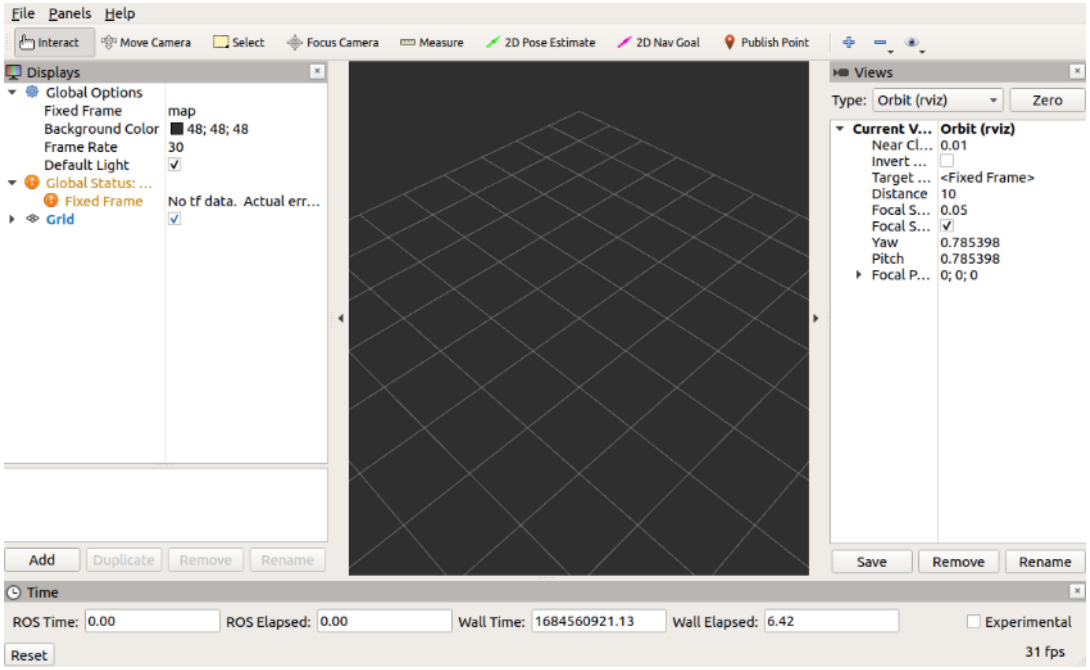


图 4.3 rviz 软件界面

（4）evo 评估工具

evo 工具是专门用来评估 SLAM 系统性能的工具，具有对比分析系统整体轨迹，计算包括平均值、中位数和均方根误差等参数在内的绝对位姿误差和相对位姿误差功能。

2. 系统实验

（1）将 KITTI 数据集转换为 bag 文件

本文基于 KITTI 数据集 07 数据段对激光 SLAM 系统进行测试，下面给出部分 07 数据段的图片。

图 4.4 KITTI 数据集 07 数据段部分原始图片 1

图 4.5 KITTI 数据集 07 数据段部分原始图片 2

图 4.6 KITTI 数据集 07 数据段部分原始图片 3

但是从官网下载的数据集文件是无法直接使用的，需要将其转换为 rosbag 才可以使用。这里要特别说明的是，rosbag 是 ros 中的数据包，可以使用 rosbag play 命令进行调用播放，并在 rviz 中显示 bag 的内容。通常用于视觉或激光 SLAM 的仿真测试。

本文基于 kitti2bag 工具对数据集进行转换，

利用 rosbag info 命令可以查看转换好的 bag 信息。

```
effun@effun-G3-3590:~/KITTI_Dataset/2011_09_30_drive_0027_sync$ rosbag info kitti_2011_09_30_drive_0027_synced.bag
path:        kitti_2011_09_30_drive_0027_synced.bag
version:     2.0
duration:    1:54s (114s)
start:       Sep 30 2011 12:40:25.56 (1317357625.56)
end:         Sep 30 2011 12:42:20.41 (1317357740.41)
size:        5.7 GB
messages:    15484
compression: none [4426/4426 chunks]
types:       geometry_msgs/TwistStamped [98d34b0043a2093cf9d9345ab6eef12e]
             sensor_msgs/CameraInfo     [c9a58c1b0b154e0e6da7578cb991d214]
             sensor_msgs/Image          [060021388200f6f0f447d0fcd9c64743]
             sensor_msgs/Imu            [6a62c6daae103f4ff57a132d6f95cec2]
             sensor_msgs/NavSatFix      [2d3a8cd499b9b4a0249fb98fd05cfa48]
             sensor_msgs/PointCloud2    [1158d486dd51d683ce2f1be655c3c181]
             tf2_msgs/TFMessage         [94810edda583a504dfda3829e70d7eec]
topics:      /kitti/camera_color_left/camera_info    1106 msgs    : sensor_msgs/CameraInfo
             /kitti/camera_color_left/image_raw      1106 msgs    : sensor_msgs/Image
             /kitti/camera_color_right/camera_info   1106 msgs    : sensor_msgs/CameraInfo
             /kitti/camera_color_right/image_raw     1106 msgs    : sensor_msgs/Image
             /kitti/camera_gray_left/camera_info     1106 msgs    : sensor_msgs/CameraInfo
             /kitti/camera_gray_left/image_raw       1106 msgs    : sensor_msgs/Image
             /kitti/camera_gray_right/camera_info    1106 msgs    : sensor_msgs/CameraInfo
             /kitti/camera_gray_right/image_raw      1106 msgs    : sensor_msgs/Image
             /kitti/oxts/gps/fix                     1106 msgs    : sensor_msgs/NavSatFix
             /kitti/oxts/gps/vel                     1106 msgs    : geometry_msgs/TwistStamped
             /kitti/oxts/imu                         1106 msgs    : sensor_msgs/Imu
             /kitti/velo/pointcloud                  1106 msgs    : sensor_msgs/PointCloud2
             /tf                                     1106 msgs    : tf2_msgs/TFMessage
             /tf_static                              1106 msgs    : tf2_msgs/TFMessage
```

图 4.7 07 数据段 rosbag 信息

由图 4.4 可以得到，该 rosbag 的点云 topic 为/kitti/velo/pointcloud。

（2）KITTI 数据集仿真测试

打开两个终端，在一个终端中输入图 4.8 中的命令启动 LeGO-LOAM 程序

```
effun@effun-G3-3590:~$ roslaunch lego_loam run.launch
... logging to /home/effun/.ros/log/a416ff96-f6cd-11ed-a308-d812654b530d/roslaun
ch-effun-G3-3590-7840.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://172.17.0.1:45639/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.13
 * /use_sim_time: True

NODES
  /
    base_link_to_camera (tf/static_transform_publisher)
    camera_init_to_map (tf/static_transform_publisher)
    featureAssociation (lego_loam/featureAssociation)
    imageProjection (lego_loam/imageProjection)
    mapOptmization (lego_loam/mapOptmization)
    rviz (rviz/rviz)
```

图 4.8 启动 LeGO-LOAM 程序命令

另一个中输入图 4.9 中的命令播放制作好的数据集 rosbag

```
effun@effun-G3-3590:~/KITTI_Dataset/2011_09_30_drive_0027_sync$ rosbag play kitti_2011
_09_30_drive_0027_synced.bag --clock --topic /kitti/velo/pointcloud
[ INFO] [1684559957.512346524]: Opening kitti_2011_09_30_drive_0027_synced.bag

Waiting 0.2 seconds after advertising topics... done.

Hit space to toggle paused, or 's' to step.
 [DELAYED]  Bag Time: 1317357625.557814   Duration: 0.000000 / 114.849475   Delay: 0.0
 [RUNNING]  Bag Time: 1317357625.557814   Duration: 0.000000 / 114.849475
 [RUNNING]  Bag Time: 1317357625.557814   Duration: 0.000000 / 114.849475
 [RUNNING]  Bag Time: 1317357625.655216   Duration: 0.097402 / 114.849475
 [RUNNING]  Bag Time: 1317357625.756221   Duration: 0.198407 / 114.849475
 [RUNNING]  Bag Time: 1317357625.763373   Duration: 0.205559 / 114.849475
 [RUNNING]  Bag Time: 1317357625.858293   Duration: 0.300479 / 114.849475
 [RUNNING]  Bag Time: 1317357625.868380   Duration: 0.310566 / 114.849475
 [RUNNING]  Bag Time: 1317357625.969482   Duration: 0.411669 / 114.849475
 [RUNNING]  Bag Time: 1317357625.970429   Duration: 0.412615 / 114.849475
 [RUNNING]  Bag Time: 1317357626.070491   Duration: 0.512677 / 114.849475
 [RUNNING]  Bag Time: 1317357626.075205   Duration: 0.517391 / 114.849475
 [RUNNING]  Bag Time: 1317357626.171448   Duration: 0.613634 / 114.849475
 [RUNNING]  Bag Time: 1317357626.178214   Duration: 0.620400 / 114.849475
 [RUNNING]  Bag Time: 1317357626.273879   Duration: 0.716065 / 114.849475
 [RUNNING]  Bag Time: 1317357626.284051   Duration: 0.726237 / 114.849475
 [RUNNING]  Bag Time: 1317357626.385068   Duration: 0.827254 / 114.849475
 [RUNNING]  Bag Time: 1317357626.385884   Duration: 0.828070 / 114.849475
```

图 4.9 播放 rosbag 命令

可以得到如下实验结果：

```
9.668722e-01 -8.260214e-03 -2.551274e-01 -4.352797e-01 1.128579e-02 -9.998822e-01 -1.039743e-02 -3.873197e-02 -2.550115e-01 -1.293230e-02 9.668516e-01 2.073963e+00
9.582301e-01 -8.731220e-03 -2.858649e-01 -5.029211e-01 1.223033e-02 -9.998705e-01 -1.045733e-02 -4.278991e-02 -2.857366e-01 -1.351675e-02 9.582129e-01 2.282266e+00
9.483130e-01 -9.274551e-03 -3.172011e-01 -5.817978e-01 1.326829e-02 -9.998575e-01 -1.043270e-02 -4.698588e-02 -3.170591e-01 -1.410218e-02 9.483010e-01 2.488293e+00
9.370861e-01 -9.879638e-03 -3.489588e-01 -6.764905e-01 1.440220e-02 -9.998425e-01 -1.036803e-02 -5.121178e-02 -3.488014e-01 -1.474151e-02 9.370808e-01 2.698701e+00
9.194986e-01 -1.511903e-02 -3.928026e-01 -8.655378e-01 1.983598e-02 -9.997716e-01 -7.952013e-03 -6.010406e-02 -3.925926e-01 -1.510349e-02 9.195885e-01 2.927428e+00
9.041632e-01 -1.573621e-02 -4.268972e-01 -9.841346e-01 2.099792e-02 -9.997504e-01 -7.620716e-03 -6.314535e-02 -4.266708e-01 -1.585432e-02 9.042681e-01 3.122914e+00
8.860810e-01 -1.604352e-02 -4.632528e-01 -1.117746e+00 2.190802e-02 -9.997334e-01 -7.281203e-03 -6.573606e-02 -4.630125e-01 -1.660068e-02 8.861963e-01 3.318643e+00
8.639501e-01 -1.349686e-02 -5.033964e-01 -1.268345e+00 2.078858e-02 -9.997445e-01 -8.873489e-03 -4.613904e-02 -5.031481e-01 -1.813115e-02 8.640100e-01 3.469423e+00
8.401961e-01 -1.356897e-02 -5.421131e-01 -1.397736e+00 2.154658e-02 -9.997328e-01 -8.370948e-03 -5.008099e-02 -5.418546e-01 -1.871392e-02 8.402640e-01 3.655937e+00
8.134590e-01 -1.389401e-02 -5.814563e-01 -1.574772e+00 2.252358e-02 -9.997172e-01 -7.622117e-03 -5.399389e-02 -5.811859e-01 -1.929676e-02 8.135420e-01 3.854537e+00
7.848343e-01 -1.430448e-02 -6.195405e-01 -1.766735e+00 2.358340e-02 -9.996988e-01 -6.793585e-03 -5.774115e-02 -6.192567e-01 -1.994271e-02 7.849353e-01 4.041926e+00
7.451375e-01 -1.873960e-02 -6.666477e-01 -1.995608e+00 2.618491e-02 -9.996565e-01 -1.167327e-03 -8.180693e-02 -6.663968e-01 -1.832593e-02 7.453721e-01 4.240332e+00
7.086263e-01 -1.874637e-02 -7.053350e-01 -2.211025e+00 2.657970e-02 -9.996467e-01 -1.351500e-04 -8.390643e-02 -7.050832e-01 -1.884337e-02 7.088742e-01 4.438320e+00
6.670831e-01 -1.854296e-02 -7.447524e-01 -2.428670e+00 2.665716e-02 -9.996441e-01 1.012175e-03 -8.573584e-02 -7.445062e-01 -1.917778e-02 6.673401e-01 4.611249e+00
6.232970e-01 -1.829771e-02 -7.817711e-01 -2.672861e+00 2.656142e-02 -9.996447e-01 2.220025e-03 -8.733712e-02 -7.815340e-01 -1.938122e-02 6.235615e-01 4.785688e+00
5.767674e-01 -1.805523e-02 -8.167089e-01 -2.950863e+00 2.628573e-02 -9.996482e-01 3.536302e-03 -8.831486e-02 -8.164855e-01 -1.942817e-02 5.770390e-01 4.957644e+00
5.175972e-01 -7.286268e-03 -8.555934e-01 -3.304660e+00 1.307516e-02 -9.999143e-01 6.054041e-04 -7.890144e-02 -8.555245e-01 -1.087366e-02 5.176481e-01 5.062646e+00
4.652488e-01 -7.144013e-03 -8.851512e-01 -3.617359e+00 1.287580e-02 -9.999163e-01 1.302562e-03 -8.075552e-02 -8.850864e-01 -1.079101e-02 4.653018e-01 5.185466e+00
4.119008e-01 -7.083442e-03 -9.112012e-01 -3.963927e+00 1.261969e-02 -9.999183e-01 2.068483e-03 -8.241235e-02 -9.111413e-01 -1.064706e-02 4.119565e-01 5.312887e+00
3.545658e-01 -1.080057e-02 -9.349687e-01 -4.300059e+00 9.124943e-03 -9.999256e-01 8.090509e-03 -7.658436e-02 -9.349866e-01 -5.662918e-03 3.546380e-01 5.371562e+00
2.993036e-01 -1.094129e-02 -9.540952e-01 -4.651016e+00 8.478458e-03 -9.999253e-01 8.807132e-03 -7.607368e-02 -9.541203e-01 -5.453251e-03 2.993740e-01 5.445352e+00
2.461276e-01 -1.115363e-02 -9.691734e-01 -5.028395e+00 7.668660e-03 -9.999248e-01 9.560027e-03 -7.546565e-02 -9.692072e-01 -5.079274e-03 2.461946e-01 5.529480e+00
1.931386e-01 -1.140636e-02 -9.811053e-01 -5.418020e+00 6.785910e-03 -9.999241e-01 1.028928e-02 -7.486237e-02 -9.811481e-01 -4.670435e-03 1.932013e-01 5.593139e+00
1.400545e-01 -1.172245e-02 -9.900744e-01 -5.828004e+00 6.031719e-03 -9.999214e-01 1.098580e-02 -7.409888e-02 -9.901254e-01 -4.433240e-03 1.401142e-01 5.633265e+00
9.354984e-02 -1.722853e-02 -9.954656e-01 -6.262892e+00 3.904568e-04 -9.998508e-01 1.726773e-02 -7.291026e-02 -9.956145e-01 1.226707e-03 9.354261e-02 5.619780e+00
5.259004e-02 -1.742293e-02 -9.984642e-01 -6.722034e+00 -2.324354e-04 -9.998475e-01 1.745931e-02 -6.834027e-02 -9.986162e-01 1.150264e-03 5.257798e-02 5.621204e+00
1.417296e-02 -1.745102e-02 -9.997473e-01 -7.170682e+00 -8.013454e-04 -9.998472e-01 1.746413e-02 -6.389424e-02 -9.998993e-01 1.048661e-03 1.415681e-02 5.624647e+00
-2.124492e-02 -1.734201e-02 -9.996239e-01 -7.636803e+00 -1.300999e-03 -9.998492e-01 1.731826e-02 -5.927675e-02 -9.997735e-01 9.325844e-04 -2.126428e-02 5.613126e+00
-5.462464e-02 -1.245901e-02 -9.993227e-01 -8.123695e+00 6.287497e-03 -9.998998e-01 1.268405e-02 -4.999695e-02 -9.993806e-01 -6.722420e-03 -3.454283e-02 5.589372e+00
-8.430687e-02 -1.216750e-02 -9.978560e-01 -8.614452e+00 6.075602e-03 -9.999024e-01 1.258399e-02 -4.725963e-02 -9.979118e-01 -6.871813e-03 -6.422666e-02 5.557273e+00
-1.128866e-02 -1.184667e-02 -9.957540e-01 -9.090906e+00 5.910945e-03 -9.999052e-01 1.243796e-02 -4.442179e-02 -9.958070e-01 -7.021292e-03 -9.120999e-02 5.522005e+00
-1.077542e-01 -1.157748e-02 -9.941102e-01 -9.624966e+00 5.908014e-03 -9.999071e-01 1.228538e-02 -4.077011e-02 -9.941601e-01 -7.197018e-03 -1.076758e-01 5.452329e+00
```

图 4.10 输出保存的位姿文件（部分）

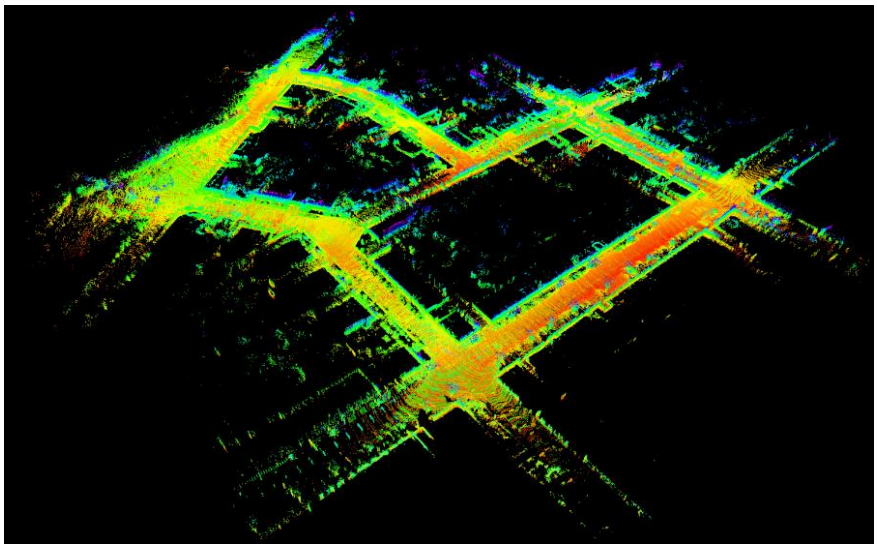


图 4.11 所建立的点云地图

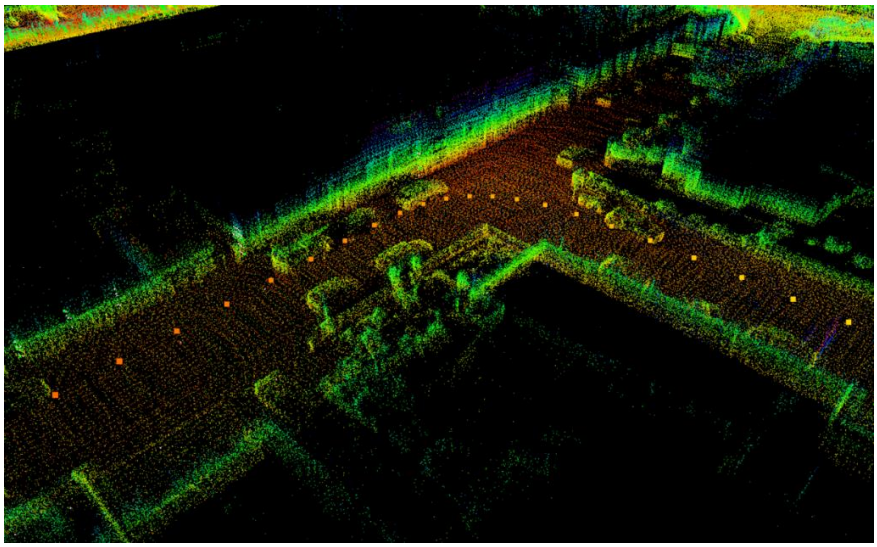


图 4.12 点云地图细节图一

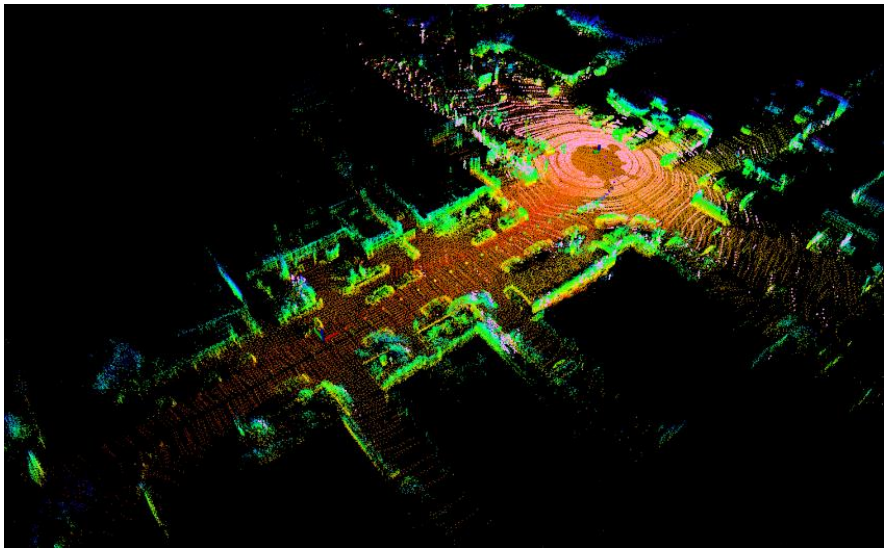


图 4.13 点云地图细节图二

3. 结果分析

（1）定位结果

基于 07 数据段的 groundtruth 真值文件，使用 evo 工具对比真值和本文系统输出的位姿值，得到全局轨迹对比图如下：

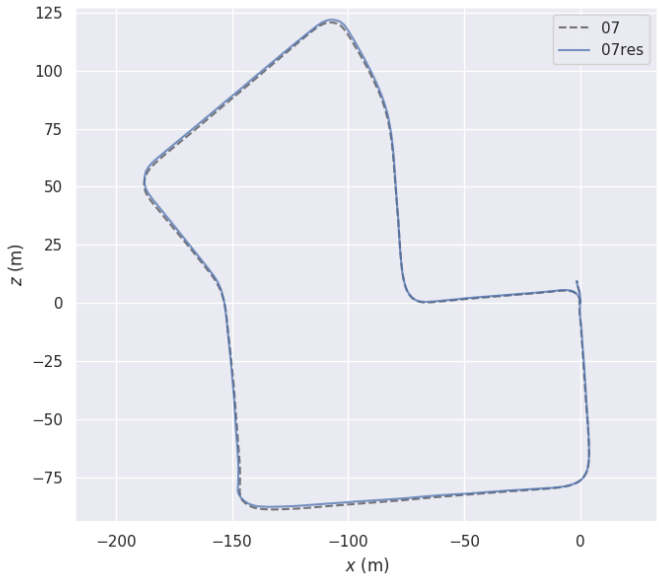


图 4.14 轨迹对比图

图 4.14 中 07 为真实轨迹，07res 为估计轨迹。可以看出，本文所设计的激光 SLAM 系统具有良好的定位效果，全局轨迹与真值几乎完全重合，漂移误差较小。

下面进一步使用 evo 工具对相对位姿误差（rpe）进行评估。

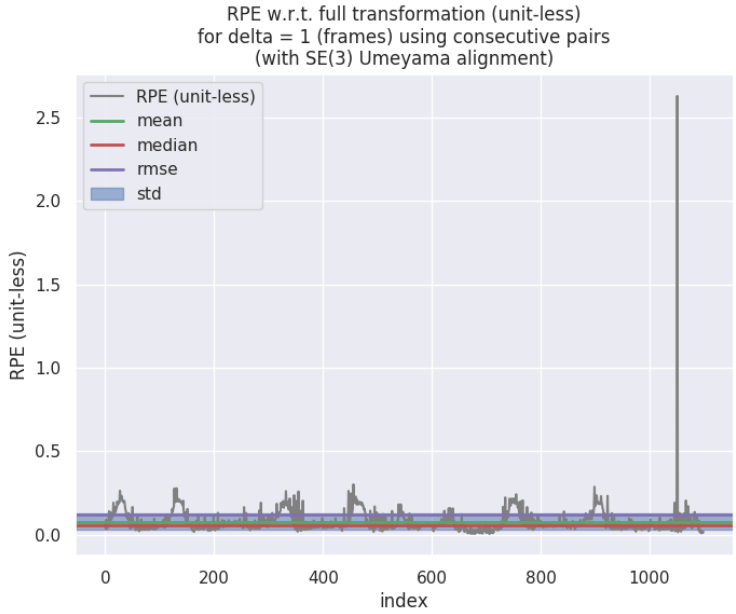


图 4.15 相对位姿误差 rpe 曲线图

```
effun@effun-G3-3590:~$ evo_rpe kitti 07.txt 07res.txt -r full -va --plot --plot_mode xyz
--------------------------------------------------------------------------------
Loaded 1101 poses from: 07.txt
Loaded 1101 poses from: 07res.txt
--------------------------------------------------------------------------------
Aligning using Umeyama's method...
Rotation of alignment:
[[ 0.99996952  0.00425492  0.00654672]
 [-0.00433062  0.99992342  0.01159337]
 [-0.00649689 -0.01162137  0.99991136]]
Translation of alignment:
[ 0.04748454 -0.95655086 -1.26693864]
Scale correction: 1.0
--------------------------------------------------------------------------------
Found 1100 pairs with delta 1 (frames) among 1101 poses using consecutive pairs.
Compared 1100 relative pose pairs, delta = 1 (frames) with consecutive pairs.
Calculating RPE for full transformation pose relation...
--------------------------------------------------------------------------------
RPE w.r.t. full transformation (unit-less)
for delta = 1 (frames) using consecutive pairs
(with SE(3) Umeyama alignment)

       max      2.630885
      mean      0.077506
    median      0.054271
       min      0.002721
      rmse      0.121923
       sse      16.351652
       std      0.094116

--------------------------------------------------------------------------------
Plotting results...
```

图 4.16 相对位姿误差的最值、均值、中值以及均方根误差信息

由图 4.15 和图 4.16 中的结果可知，在 1101 帧中，仅有 1000 帧后的某几帧出现了位姿的漂移，且峰值在 2.63m。估计位姿的整体误差均值为 0.0775m，中值为 0.00543m，均方根误差为 0.1219m。考虑到 07 数据段是大规模的室外场景，因此可以认为本文所设计的激光 SLAM 系统在室外环境下具有较好的定位精度。

（2）建图结果

图 4.11 给出了全局稠密点云地图构建结果，图 4.12 和图 4.13 分别展示了部分点云地图的细节，可以看出，所建立的 3 维点云地图对真实场景的还原度很高，可以认为室外建图效果较好，能够建立全局一致的稠密点云地图。这对于后续机器人导航具有重要意义。

五. 总结

本文基于主流的 LeGO-LOAM 框架，完成了激光 SLAM 系统设计。为使用 KITTI 数据集对系统性能进行验证，针对其特性对代码进行修改，并为系统添加了输出保存 KITTI 格式位姿的功能。最后基于 KITTI-07 数据段对整个系统进行了测试，验证了所设计的激光 SLAM 系统切实可行，能够在室外空旷环境下稳定运行并具有一定的精度。

本文完成的主要工作有：（1）对五种主流激光 SLAM 算法进行论述分析，确定本文基于 LeGO-LOAM 框架进行系统设计与实验；（2）对 SLAM 问题进行详细描述，给出了基于列文伯格-马夸尔特方法的 SLAM 问题求解过程；（3）详

细分析了 LeGO-LOAM 算法中点云分割与聚类、特征提取与匹配、激光里程计和位姿优化与建图的原理，并基于 LeGO-LOAM 框架完成了激光 SLAM 系统设计，为其添加位姿保存功能；（4）搭建系统实验所需要的软件环境，包括安装 Ubuntu18.04 系统、ROS-melodic 系统以及安装 Gtsam、PCL、OpenCV 和 Python 等在内的各版本依赖库；（5）完成了 KITTI 数据集 rosbag 的制作；（6）利用制作好的 rosbag 对所设计的激光 SLAM 系统进行了测试，验证了系统具有较高的精度和稳定性。

本文的主要改进点为：（1）针对 KITTI 数据集录制时使用的激光雷达不同修改代码参数，使得 LeGO-LOAM 算法能够适配该数据集进行测试；（2）在深入理解代码原理的基础上为 LeGO-LOAM 添加位姿保存功能，以便于和真值进行对比，验证系统性能。

本文依然存在一些不足之处：（1）受限于任务量和时间，本文对激光 SLAM 的对比分析只局限于文字，没有理解并运行多个算法框架进行横向对比。（2）受限于硬件设备，没有下载更多的数据段对系统进行测试。

参考文献

[1] Leonard J J, Durrant-Whyte H F, Cox I J. Dynamic map building for an autonomous mobile robot[J]. International Journal of Robotics Research, 1992, 11(4): 286-298.

[2]徐健伟. 面向复杂环境的移动机器人建图与定位[D]. 合肥:合肥工业大学, 2021.

[3]W. Hess, D. Kohler, H. Rapp and D. Andor. Real-time loop closure in 2D LIDAR SLAM. IEEE International Conference on Robotics and Automation [C], IEEE, 2016, 1271-1278.

[4] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time [C]. Proceedings of Robotics: Science and Systems, 2014,1561-1569.

[5] T. Shan and B. Englot. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain [C]. IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2018: 4758-4765.

[6] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti and D. Rus,. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping [C]. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, 5135-5142.

[7]高翔，张涛. 视觉 SLAM 十四讲：从理论到实践[M]. 北京：电子工业出版

社, 2017.
[8]王义林. 地面无人平台自主导航避障系统的研究与实现[D]. 哈尔滨:哈尔滨工业大学,2020.
[9]I. Bogoslavskyi and C. Stachniss. Fast Range Image-based Segmentation of Sparse 3D Laser Scans for Online Operation. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems[C]. IEEE, 2016: 163-169.
[10]刘晓波,张明明,涂俊超,等. 基于广度优先搜索的小波聚类算法[J]. 振动与冲击,2016,35(15):178-183.
[11]陈文浩. 基于 LiDAR 与 IMU 融合的 SLAM 算法研究及其在校园巡检机器人上的应用[D]. 武汉:华中师范大学,2020.
[12]陈文浩,刘辉席,杨林涛等. 基于 IMU 紧耦合的 LeGO-LOAM 改进算法研究[J]. 计算机应用研究,2021,38(4):1013-1016.