# 论文推荐

《LOAM\_Lidar Odometry and Mapping in Real-time》

《Low-drift and real-time lidar odometry and mapping》

本文作为三维激光SLAM的经典之作，是众多入行激光SLAM的必修课，为激光雷达和IMU融合（没用到）的算法。Ji Zhang 曾经在网上公开过有关LOAM的算法，但在很快的时间里，代码就下线了。但借助github平台强大的力量和开发者的激情，众多激光SLAM的研究者纷纷在github上放出自己开发的的LOAM算法版本，不同于Zhangji使用的Hokuyo UTM-30LX激光雷达传感器，研究者更多依赖的是Velodyne公司开发的相关激光雷达，所以目前网上最为出名的LOAM算法版本大多称为LOAM\_Velodyne。其中最出名的是github用户Leonid Laboshin用户和Daobilige Su分别开发的版本。后续的众多研究者在上面继续开发，出现了一系列优秀的有关LOAM的进一步开发。在2018年4月，港科大秦通等人，对LOAM-Velodyne进行再次开发，发布了A-LOAM算法版本。

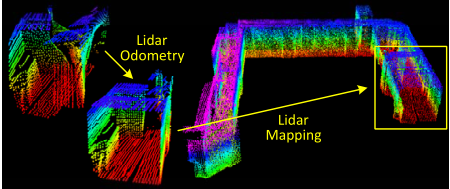
# 创新点

（1）在前端里程计特征提取和点云配准方面，提出了新颖的ICP算法，提取边缘点与平面点作为关键特征点，用点对线，点对面进行点云帧间配准，提高了配准精度。

（2）将同时定位与建图分为两部分进行，第一部分为高频率（10HZ）低精度的里程计odometry过程，另一个是低频率（1HZ）高精度的建图mapping过程，二者结合可实现低漂移、低计算量、高精度的SLAM。

# 思想

1、一种算法是在高频但低保真的情况下执行里程计来估计激光雷达的速度2、另一种算法以较低数量级的频率运行，用于点云的精细匹配和配准。

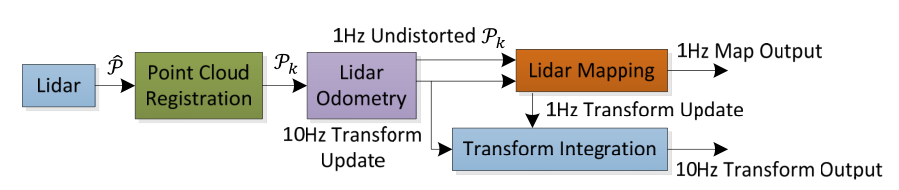


图左侧为运用 **点云运动畸变去除的激光雷达里程计**，

右图为 **激光雷达建图**

1、该方法的目标是使用移动的2轴激光雷达进行运动估计和建图。由于激光点是在不同的时间接收的，因此由于激光雷达的运动(如左边的激光雷达云所示)，点云中会出现变形。该方法将问题分解为两个并行运行的算法。里程计算法估计激光雷达的速度并校正点云中的变形，

2、然后，建图算法匹配并配准点云以创建地图，两种算法的结合保证了问题实时求解的可行性。



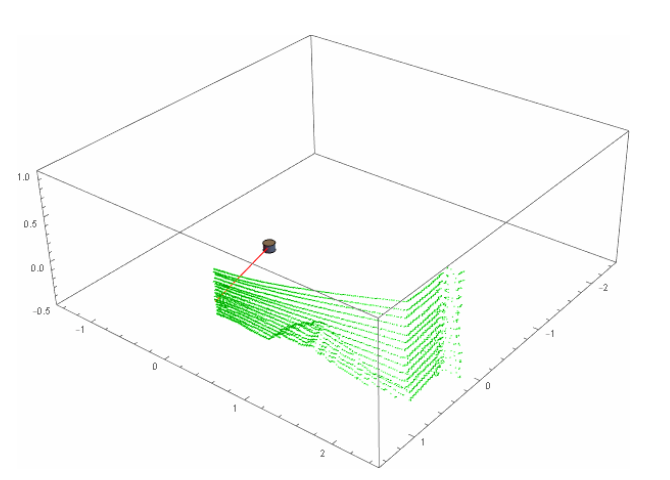
1. 首先是获得激光雷达坐标系下的点云数据P^，然后把第k次扫描获得的点云组成一帧数据Pk。然后将Pk在两个算法中进行处理，也就是上面Liar Odometry节点和Lidar Mapping节点。
2. Liar Odometry节点的作用是获取两帧连续点云数据间的运动，估计出来的运动用于去除Pk中的运动畸变。这个节点执行的频率为10Hz，作用相当于scan-to-scan匹配获得粗糙的运动估计用于去除匀速运动造成的运动畸变，并将处理后的结果给了Lidar Mapping节点做进一步处理。
3. Lidar Mapping节点使用地图去匹配和注册没有畸变的点云数据以1Hz的频率，作用相当于scan to map。最后由Transform integration节点接收前面两个节点输出的Transform信息并将其进行融合处理以活动频率为10Hz的Transform信息即里程计。

## Point Cloud Registration（scanRegistration.cpp）

### （1）点云预处理

传感器获得的数据可能在一些点的坐标中为NaN（不是数）值, 一个NaNs表明测量传感器距离到该点的距离值是有问题的，可能是因为传感器太近或太远，或者因为表面反射。那么当存在无效点云的NaNs值作为算法的输入的时候，可能会引起很多问题，使用PCL自带的函数以及自己写的过近点去除函数进行去除。（去除NaN点和INF点）

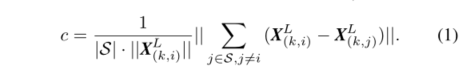
velodyne 16线激光雷达返回的点云数据是按照先上下，后左右的顺序，如下图所示。



因此在程序中需要先将其按照线号重新排列。velodyne 16雷达每次返回的数据称为一帧（sweep），一帧由16条线组成（每条线称为一个scan），将他重新排列好。

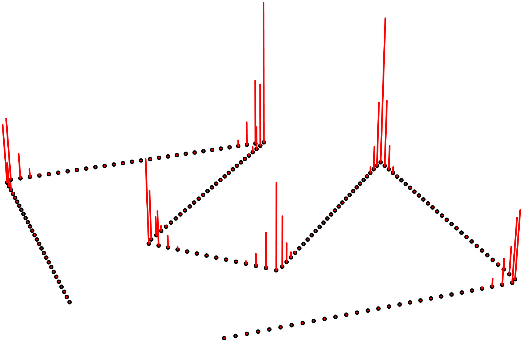
### （2）提取特征点

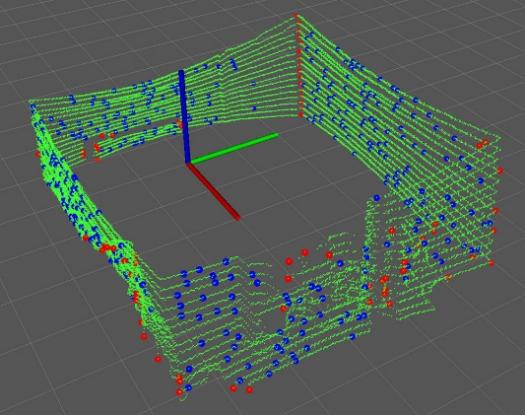
选择位于边缘和平面上的特征点。设I是PK中的一个点，I∈PK，S是激光扫描仪在同一扫描中返回的I的连续点的集合。由于激光扫描仪以顺时针或逆时针的顺序生成点云，因此S在I的两边各包含一半的点，两点之间的间隔为0.25◦。如何选取边缘点和平面点呢？利用点i周围连续几个点集合S用于求曲率c。

****

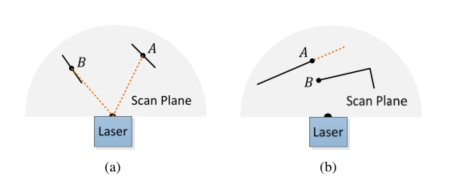
基于曲率c值对扫描中的点进行排序，然后选择c最大的(即边缘点)和最小的c(即平面点)的特征点。

如果点X在一条直线上而且直线上的点都是均匀分布的，那么不管这条直线方向如何，X的前5个点和后5个点的平均值刚好就是X（此时c=0）。上面公式就是在计算平均点与中间点X的差距，点的排列越平直差距越小，反之点排列越弯曲差距越大c cc的值就越大。下面给出两个例子来更好地展示曲率的含义。左图为平面点云的例子，其中有的点在直线上有的位于边角上，我们计算各点处的曲率。为了直观地展示曲率的大小，我用直线表示在各个点上，直线高度与曲率c成正比，如右图所示。越尖锐的点曲率越大，在直线上的点曲率则是0。





选取技巧：在选择特征点时，我们希望避免选择已经选择的点的临近点，或局部平面上大致平行于激光束的点(下图(a)中的点B)，因为这些点太近了，很不可靠。此外，我们还希望避免位于遮挡区域边界上的点(下图(b)中的点A)。



实线段表示局部平面。点A位于与激光束成一定角度的平面上(橙色虚线段)，点B位于大致平行于激光束的平面上。我们将B视为不可靠的激光点，则不选择它作为特征点。

实线段是激光雷达的可观察对象。点A位于遮挡区域(橙色虚线段)的边界上，那么可以视为边缘点。但是，如果从不同的角度观察，被遮挡的区域可能会改变并变得可见，则不会将A视为明显的边缘点，也不会选择它作为特征点。

那么特征点的选取规则如下：

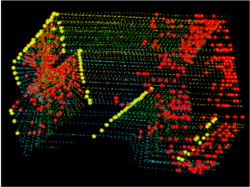
（1）从最大曲率c值开始选取边缘点，从最小c值开始选取平面点；

（2）所选边缘点或平面点的数量不能超过子区域的最大值；

（3）所选点的临近点不剋被选择

（4）不能位于与激光束大致平行的平面上，也不能位于被遮挡区域的边界上。

选取特征点的实例如下图所示：在长走廊场景中提取特征点



黄色：边缘点edge points 红色：平面点planar points

**曲率计算完成后进行特征分类，提取特征点有几点原则：**

1.为了提高效率，每条扫描线分成6个扇区，在每个扇区内，选取极大边线点（2个）和次极大边线点（20个）

2. 寻找曲率最小的4个点，作为极小平面点，剩下未被标记的点，全部作为次极小平面点。

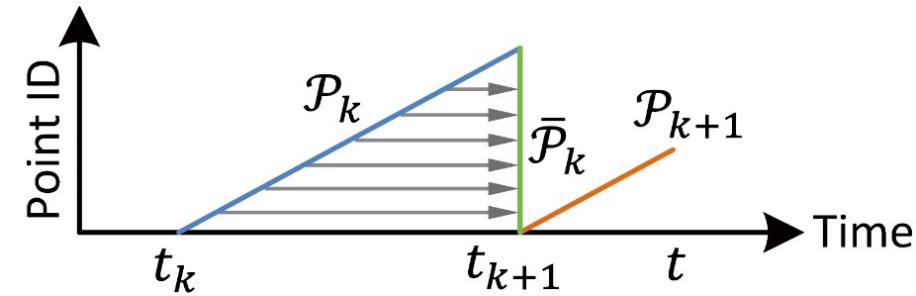
3. 对于次极小平面点，由于点数太多，采取体素栅格滤波降采样的方法避免过多聚堆。

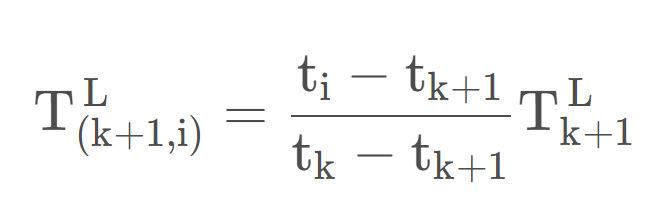
## Lidar Odometry (laserOdometry.cpp)

### 运动补偿（畸变矫正）

LOAM还解决了运动畸变问题（demotion或者distortion correction）。运动畸变产生的原因就是激光雷达在采集数据的过程中是处于运动状态的。如果激光雷达的扫描频率很高，比自身的运动快得多，我们可以假设畸变很小从而忽略。但是，大多数雷达的频率都不是非常高，以velodyne 16线雷达为例，常用的频率是10Hz（最快也不过20Hz），假如机器人或者无人车的运动速度是1m/s（这算慢的了），在完成一次360°扫描的过程中雷达的位置已经改变了10cm，这就不能再忽略了。LOAM解决运动畸变的方法比较简单，就是根据每个点的相对时间进行补偿。雷达扫描一帧的时间是固定的，可以得到每个点的采集时刻，将所有点都统一到同一时刻，这里选择的是每完成一帧扫描的末尾时刻，如下图所示。t k

就是一帧扫描开始的时刻，t k + 1就是完成一次扫描的时刻，对于我们的例子t k + 1 − t k = 100 ms水平的箭头表示将所有点都投影到t k + 1 时刻。P k 就是这一帧扫描生成的点云，显然不同的点具有不同的时间戳。





### （2）点云配准

再提取了特征点之后，我们需要做的就是特征匹配了。这里使用的使scan-to-scan的方法来实现帧与帧之间的特征匹配。

已知第k次扫描的点云为P*k*，而提取的边缘点集合记为: E*k*，提取的平面点记为H*k*。

已知第k＋1次扫描的点云为P*k+1*，而提取的边缘点集合记为: E*k-1*，提取的平面点记为H*k-1*。

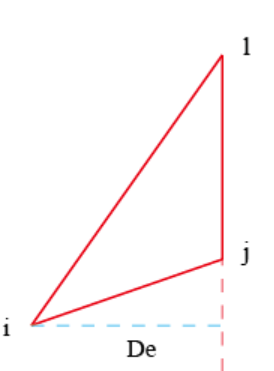
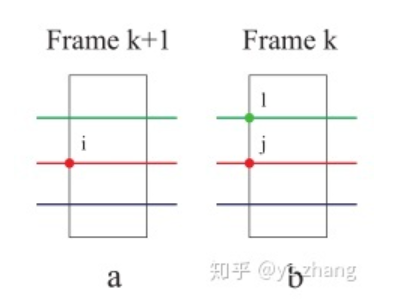
我们想要得到的是P*k+1*和P*k*之间的变换关系，也就是E*k+1*和E*k*以及H*k+1*和H*k*之间的关系。由于雷达自身在k和k＋1时刻过程中是运动的，所以，我们每个点对应的姿态变换矩阵都应该得到一定的修正。为了方便处理，我们将所有的点重投影到每一帧的初始时刻，这样在这一帧中的所有点都可以得到对应的姿态变换信息。

我们将重投影到每一帧初始时刻的平面点和边缘点记为: 和 。这样的话就可以进行后续的优化了。

**边缘点匹配**

已知信息和E*k*。

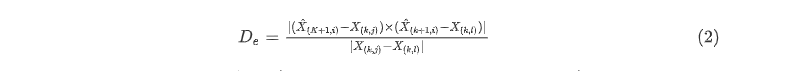
我们知道，边缘点就是三维结构中线所构成的点，这样的话，就是求点到线的最近距离。需要在*Ek*。中找到一条线来求解最近距离:



从中选取一个点i，在Ek。中选取与i最近的点j，以及和点j相邻扫描线中最近的点l，这样的目的是防止i,j,l三点共线而无法构成三角形。

选取最近点的算法使用的是kd-tree的最近邻搜索。





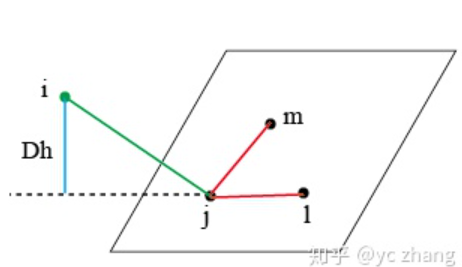
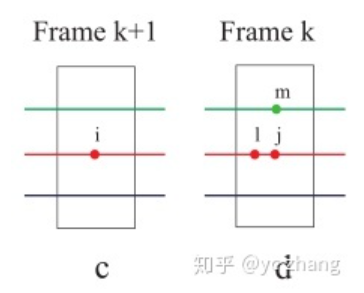
我们知道，分子叉乘球出来的是**j**和**il[[1]](#footnote-0)**构成的平行四边形的面积，而分母则是**jl**构成的底。

因此，我们就构建了边缘点的优化方程。

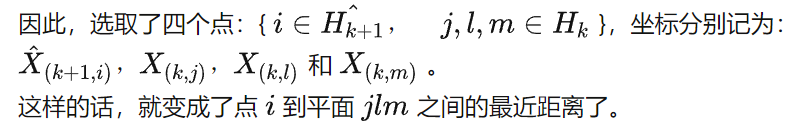
**平面点匹配**

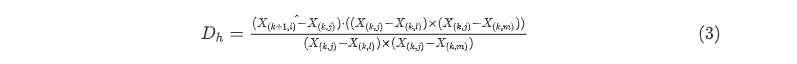
已知信息和Hk 。

平面点的匹配起始和边缘点的匹配类似，同样的是寻找两帧之间的对应关系，我们知道平面点的话，就是要求点到平面的距离，这样的话，就需要在H*k*中找到一个对应的平面。



从中寻找一个点i，从H*k*中找寻与点i最近的点l，并找到与点Ⅰ相同激光扫描束的最近点j，然后，找寻相邻帧中与点j最相近的点m，这样的话，就可以找到一个不共线的，能构成一个平面的三个点。





分子为构成的三维物体的体积，分母为地面构成的平行四边形的面积，则高 体积面积Dh=体积/面积 。因此，我们就得到了平面点的优化方程。

LOAM使用了点到直线和点到平面来计算距离，这就意味着他适用于结构化比较好的室内环境，因为在室内环境中充满了平直的墙壁、地面、家具、平直的转折角。对于室外环境就没这么理想了，可能充满了树叶、石块、车辆等等不规则的物体，其直线和平面的假设就不太准确了。

## Lidar Mapping

LOAM缺少闭环检测，它建图的主要目的还是为状态估计服务的。论文中说建图的频率与状态估计不同，比它要慢。扫描一帧就对地图做一次更新，如果用velodyne 16雷达就是10Hz，而论文中使用了简陋的自制激光雷达，建图频率只能做到1Hz。

建图的过程就是不断地把匹配好的点云堆积在一起的过程，其中的思路与状态估计有些类似，但是有很多地方不一样。特征点的定义和使用与前面状态估计的一样，但是数量更多了，多了10倍。为什么多了10倍大概是因为建图的频率慢了10倍吧。在寻找对应特征点时，将地图中已有的点云Q k按照10立方米的格子存储。至于为什么是10立方米我也不知道。使用已经粗略估计出来的单帧点云 Qk+1它是相对于世界全局坐标系的。凡是与Q k + 1 有交集的方格，从Q k取出位于这些方格中的点，再存入一个KDtree中。然后，再针对不同的特征点找它同类的那些点，平面点同理，各找各妈。在寻找时限制搜索范围，只找一定半径范围内的。由于现在特征点的数量增加了十倍，再挨个计算距离太慢了，此时作者采用了计算特征向量的方法。为什么用特征向量了呢？这让我们想起了另一种点云匹配方法——NDT方法。

然后在直线上取两个点利用前面的公式计算角点到直线的距离，在平面上取三个点计算平面点到平面的距离。后面的操作跟前面的一样，也是利用LM方法得到最优的变换，利用新得到的最优变换，将最近的一帧单帧点云叠加到地图点云之上，就完成了建图。新得到的最优变换与之前粗略估计出来的变换一般是不同的，新的变换由于使用了更多点参与计算，应该更接近真实，所以用新的变换作为以后变换的基础。

1. 加粗表示向量 [↑](#footnote-ref-0)