## 一、学习记录

程序入口是loam\_velodyne.launch。

Launch文件里最常用的标签是node，一般写法是：

<node pkg=”包名” type=”node类型” name=”node名字” output=”screen” respawn=”是否自动重启” />

此外还用到arg标签：

<arg name=”rviz” default=”true” />

还有group标签，经常用来写rviz

好了，分别看这几个node吧。

由于代码风格问题，loam定义了大量的全局变量。我们跳过这些东西，直接看main函数吧。

### 1、scanRegistration.cpp

先来看传感器入口，scanRegistration

ros::init(argc, argv, **"scanRegistration"**);  
ros::NodeHandle nh;

定义node名称和句柄。如果想让消息不带node名，句柄就不要带~

下面是消息的发布和订阅：scanRegistration订阅点云和imu两个消息；发布/velodyne\_cloud\_2、/laser\_cloud\_sharp、/laser\_cloud\_less\_sharp、/laser\_cloud\_flat、/laser\_cloud\_less\_flat、/imu\_trans六个消息。

订阅的写法：

nh.subscribe<sensor\_msgs::PointCloud2>(“velodyne\_points”, 10, handler);

nh.advertise<sensor\_msgs::PointCloud2>(“velodyne\_points”, 10);

里面主要是两个回调函数imuHandler和laserCloudHandler

imuHandler其实也没干啥，就做了坐标转换，然后把加速度在世界系下做了个积分。

重点是laserCloudHandler。

流程是分线束、运动补偿、计算曲率、计算特征点、发布消息。

scanRegistration结束了，重点算法是曲率的计算以及特征点的计算。

### laserOdometry.cpp

未完待续

## 重点算法

### 曲率计算

由当前点出发，指向10个最近的邻居，得到10个向量。这10个向量的和向量的模，就是“曲率”。

## 特征点的计算

将360度的区域六等分，每个区域找到2个sharp和18个less\_sharp，以及4个flat和剩余的less\_flat。

未完待续

## 三、