amcl #即时定位

│ ├─ amcl.launch.xml #即时定位核心启动文件

│ ├─ astra\_amcl.launch.xml -&gt; amcl.launch.xml #使用astra进行定位

│ ├─ asus\_xtion\_pro\_amcl.launch.xml -&gt; amcl.launch.xml

#使用asus\_xtion\_pro进行定位

│ ├─asus\_xtion\_pro\_offset\_amcl.launch.xml -&gt; amcl.launch.xml

#使用asus\_xtion\_pro\_live进行定位

│ ├─ kinect\_amcl.launch.xml -&gt; amcl.launch.xml #使用kinect进行定位

│ ├─ r200\_amcl.launch.xml #使用r200进行定位

amcl是移动机器人二维环境下的概率定位系统。它实现了自适应（或kld采样）的蒙特卡罗定位方法，也就是在已知地图中使用粒子滤波器方法计算机器人的姿态。

粒子滤波很粗浅的说就是一开始在地图空间很均匀的撒一把粒子，然后通过获取机器人的motion来移动粒子，比如机器人向前移动了一米，所有的粒子也就向前移动一米，不管现在这个粒子的位置对不对。使用每个粒子所处位置模拟一个传感器信息跟观察到的传感器信息（一般是激光）作对比，从而赋给每个粒子一个概率。之后根据生成的概率来重新生成粒子，概率越高的生成的概率越大。这样的迭代之后，所有的粒子会慢慢地收敛到一起，机器人的确切位置也就被推算出来了。

### 发布的话题：

1、amcl\_pose：geometry\_msgs::PoseWithCovarianceStamped，后验位姿+6\*6的协方差矩阵（xyz+三个转角）

2、particlecloud：geometry\_msgs::PoseArray，粒子位姿的数组

3、一个15秒的定时器：AmclNode::checkLaserReceived，检测上一次收到的激光雷达数据至今是否超过15秒，如超过则报错。

### 发布的服务：

1. global\_localization：&AmclNodee::globalLocalizationCallback，在没有给定初始位姿的情况下，在全局范围内初始化粒子位姿，该Callback调用pf\_init\_model，然后调用AmclNode::uniformPoseGenerator在地图的free点随机生成pf->max\_samples个粒子、
2. request\_nomotion\_update：&mclNode::nomotionUpdateCallback没有运动模型时也更新粒子群
3. set\_map：&AmclNode::setMapCallback
4. dynamic\_reconfigure::Server动态参数配置

### 订阅话题

1. scan\_topic：sensor\_msgs::LaserScan，AmclNode::laserReceived，订阅tf，转换到odom\_frame\_id
2. initialpose：AmclNode::initialPoseReceived，订阅rviz中给的初始位姿
3. Map：AmclNode::mapReceived，在use\_map\_topic 时订阅，否则requestMap();

### amcl节点

输入：激光地图；初始化粒子滤波器的均值（初始位姿）协方差（所有粒子的后验位姿），用于初始化粒子（粒子集）；里程计信息；激光数据；tf坐标变化

输出：在地图中的估计姿势（粒子后验位姿以及协方差矩阵，xyz和三个转角）、粒子计

在源码中，amcl\_node.cpp文件实现了amcl节点功能，主要流程为：

1：获取激光地图：调用 handleMapMessa（）

2：获取用于初始化粒子滤波器的均值（初始位姿）协方差（所有粒子的后验位姿）初始化粒子集、里程计信息、激光数据、tf坐标变化

3：更新机器人位置坐标bindLaserRecieved（）

a：获取激光数据对应于里程计中的坐标

c：通过里程计的运动模型（差分型双轮机器人）更新现有的每一个粒子的位姿（得到当前时刻的先验位姿）

odom->updateAction()

d：根据当前激光数据更新各里程计对应的权值weights

e：获取权值最高的坐标点进行聚类（得到均指和方差，把相近的很多粒子融合成一个粒子），然后将新粒子发布到全局坐标系下（地图坐标系）。

f：遍历所有粒子簇，找出权重均指最大的簇，这个平均位姿即为当前机器人的位姿（后验位姿）

g：更新位姿、粒子集、协方差矩阵