

上图为cartographer官网给出的运算流程，也是cartographer算法的框架图。

按着框架来看程序的话，首先是传感器数据，这些统一在源码的sensor包里，包括imu，odometry，激光，fixed fram

1. 首先是激光数据，cartographer对他进行了体素滤波，更确切点是自适应体素滤波，体素滤波是为了后面的自适应体素滤波提供了一个接口，体素滤波在sensor/internal里面(具体就是给定一定值的激光范围，然后给定这个范围内的激光点云数据的最小值，通过不断缩小体素范围长度，达到这个要求，从而得到滤波的效果，具体数值在配置文件里面)。

滤波之后的激光数据一是被送入了局部slam里面进行扫描匹配，具体文件在mapping/internal/2d/scan\_matching/real time CSM，

二是被送入了位姿插值器里面，用来进行推测位姿

1. 里程计数据，并没有进行特别的处理，直接就被位姿插值器调用了，它的定义再sensor包里面，被调用是在mapping/pose\_extrapolator里面。
2. Imu数据，同样定义是在sensor包里面，后面经过mapping/imutracker调用，在imutracker里面通过计算重力在各个方向上的分量，计算出各轴的加速度。然后把数据给到mapping/pose\_extrapolator。
3. Fixed frame是为了在计算姿态图时，给定一个标准，使姿态图可以固定下来。

通过对传感器数据的调用，通过位姿插值器mapping/pose\_extrapolator,来融合imu，里程计和激光数据，因为imu和里程计的频率会比激光数据高，所以在一段时间内只用imu和里程计数据来估算位姿，当有激光数据传进来时，在对位姿进行估计。

然后就是local slam部分，在mapping/internal/2d/local\_trajectory builder里面，首先就是调用激光数据，进行scan match，策略就是real time CSM 和 ceres scan match在mapping/internal/2d/scan\_matching里面，通过扫描匹配构建子地图。

接着就是后端优化的过程，这部分内容在mapping/internal/global\_trajectory builder里面，首先是将local slam所生成的子地图添加进来，通过扫描匹配策略fast CSM,fast CSM就是分支定界算法，在mapping/internal/2d/scan\_matching里，scan match完成之后，通过make constrain文件创建约束，在pose graph 2d里与imu数据，里程计数据一起构建约束形成稀疏姿态图，进行求解，从而得到地图。



上面图片里的几个函数，都是在cartographer\_ros包里面的，按顺序排列的，通过一步步调用，把程序从cartographer\_ros手里转到cartographer手里。

Map\_builder\_bridge,转到cartographer里面的mapping/map\_builder里面，从这个函数正式转入到local\_trajectory\_builder和global\_trajectory\_builder里面，也就是上面描述的cartographer结构，可以从global\_trajectory\_builder开始看起。

cartographer概述

cartographer主要包括两个部分，一个是前端匹配，另一个就是后端优化工作，首先前端匹配的过程就是创建submap的过程，通过添加一系列的经过体素滤波的scan，和imu（2D可以没有，3D必须要有）、里程计数据信息进行位姿融合，通过scan match策略添加到submap当中去，这里的scan match策略就是CSM（但是在程序当中用到了两个,real time CSM和ceres scan match。real time CSM为ceres scan match提供一个很好的初值，之后用ceres scan match 进行优化）。在submap被创建的时间内，cartographer认为误差是很小的，但是当大量的submap被创建完成时，就会有误差积累，这时引入后端回环检测，进行优化。

后端检测，这个过程是一直在运行的，也即是当一旦有laser scan加入进来时，就会进行回环检测。前端得到的submap会不断加入到后端中，当一个scan加入进来时，cartographer就会考虑回环检测，进行scan match，这里的scan match策略就是FAST CSM，也就是分枝定界算法，这个算法通过将栅格分为几层深度（通俗来讲就是分了好几层分辨率），通过先匹配低分辨率再匹配高分辨率，大大的缩短了回环检测的时间。一旦建立了回环，就会进行优化，优化的方式，就是构建位姿图，通过将imu数据、里程计数据、激光数据、submap数据形成各自的约束，建立一个多边形回环，建立环上的节点，通过稀疏位姿图进行优化，优化的方式是建立一个非线性最小二乘方程进行优化（这些优化的结果全部在ceres库里解决），最终得到全局地图。