两点注意：

1、这个[Cartographer](https://github.com/googlecartographer/cartographer)的ROS包使用了[TF2](http://wiki.ros.org/tf2)，因此所有的坐标系的名称（包含下划线的小写字符）都是唯一的，不能有前缀或斜杠，参考常用的坐标系文档[REP\_105](http://www.ros.org/reps/rep-0105.html)。

2、这个[Cartographer](https://github.com/googlecartographer/cartographer)的ROS包的topic名字作为基本名字（参考[ROS names](http://wiki.ros.org/Names)），这意味着需要用户进行remap或将它们放入命名空间中。

以下是此包的顶层选项，所有的这些都需要在Lua的配置文件中指定：

**map\_frame**

用来发布子地图的ROS坐标系ID，位姿的父坐标系，通常是“map”。

**tracking\_frame**

SLAM算法跟随的坐标系ID，如果使用了IMU，应该是IMU的位置，尽管它可能转动了，常用的选择是“imu\_link”。

**published\_frame**

作为发布位置的子坐标系的ROS坐标系ID，对于“odom”例程，如果系统的其他地方提供“odom”坐标系，那么“odom”在*map\_frame*中的位置将会被发布，否则，设置为“base\_link”即可。

**odom\_frame**

“*provide\_odom\_frame*”为真时使用，位于“published\_frame ”和“map\_frame”之间，用来发布本地SLAM结果（非闭环），通常是“odom”。

**provide\_odom\_frame**

如果使能，这个局部的，非闭环的，连续的“odom\_frame ”在“map\_frame”中的位置将被发布。

**use\_odometry**

如果使能，将订阅[nav\_msgs/Odometry](http://docs.ros.org/api/nav_msgs/html/msg/Odometry.html)类型的topic“odom”，这种情况下，必须提供里程计，而且里程计的信息将被包含在SLAM中。

**num\_laser\_scans**

订阅的laser scan topics的个数，对于单个激光，订阅[sensor\_msgs/LaserScan](http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html)类型的“scan”topic，对于多个激光，topics为“scan\_1”, “scan\_2”...。

**num\_multi\_echo\_laser\_scans**

订阅多回波技术laser scan topics的个数，对于单个激光，订阅[sensor\_msgs/MultiEchoLaserScan](http://docs.ros.org/api/sensor_msgs/html/msg/MultiEchoLaserScan.html)类型的“echoes”topic，对于多个激光，topics为“echoes\_1”, “echoes\_2”...。

**num\_subdivisions\_per\_laser\_scan**

接收到的（多回波）laser scan中分离出来的点云的个数，分割scan能够保证在激光设备移动的过程中scans不弯曲变形。有相应的轨迹生成器选项可以将分离的scans累积到点云中，用来进行scan matching。

**num\_point\_clouds**

订阅的点云topics的个数，对于单个测距仪，订阅[sensor\_msgs/PointCloud2](http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html)类型的“points2”topic，对于多个测距仪，topics为“points2\_1”, “points2\_2”...。

**lookup\_transform\_timeout\_sec**

使用[tf2](http://wiki.ros.org/tf2)进行转换搜素的超时时间的秒数。

**submap\_publish\_period\_sec**

发布submap位置的间隔秒数，如0.3s。

**pose\_publish\_period\_sec**

发布位置的间隔秒数，如5e-3对应200Hz。

**trajectory\_publish\_period\_sec**

发布轨迹标记的间隔秒数，如30e-3对应30ms。

## **二、调试**

很不幸，对cartographer进行调优很困难。这个系统有相当多的相互影响的参数。

这个调试说明试图解释在具体例子上的一些原则。

### **2.1 两个系统**

Cartographer可以看做是两个分离的，却相关的系统。一个叫做局部SLAM（有时候也叫作前端），它的作用是构建一个局部一致的submaps 集，但是它会随时间的推移而变化，有关的它的大部分选项都能在相对应的[trajectory\_builder\_2d.lua](https://github.com/googlecartographer/cartographer/blob/aba4575d937df4c9697f61529200c084f2562584/configuration_files/trajectory_builder_2d.lua)或[trajectory\_builder\_3d.lua](https://github.com/googlecartographer/cartographer/blob/aba4575d937df4c9697f61529200c084f2562584/configuration_files/trajectory_builder_3d.lua)中找到。

另一个系统叫做全局SLAM(有时候也叫作后端)，它在后台线程中运行，主要作用是找到闭环的约束条件，通过扫描submaps 来完成scan-matching。也结合其他的传感器数据以获得更高水平的视图，确定最一致的全局解决方案。在3D模式下，还会尝试找到重力的方向，有关的大部分选项都能在[pose\_graph.lua](https://github.com/googlecartographer/cartographer/blob/aba4575d937df4c9697f61529200c084f2562584/configuration_files/pose_graph.lua)中找到。

更抽象一点来说，局部SLAM是生成良好的submaps ，全局SLAM是把它们联系在一起。

### **2.2 调试局部SLAM**

对于这个例子，我们将从我们的测试数据集中找到cartographer commit [aba4575](https://github.com/googlecartographer/cartographer/commit/aba4575d937df4c9697f61529200c084f2562584) 、cartographer\_ros commit [99c23b6](https://github.com/googlecartographer/cartographer_ros/commit/99c23b6ac7874f7974e9ed808ace841da6f2c8b0)和b2-2016-04-27-12-31-41.bag。

在我们的初始配置中，我们看到在这个包的早期就有一些滑动，这是因为这个包经过了德国博物馆的一个斜坡，这违反了平面地板的二维假设。在激光扫描数据中可以看到相矛盾的信息被传递给SLAM。但这一下滑也表明，我们对点云的匹配程度过高，而且对其他传感器的忽视也相当强烈。我们的目标是通过调优来改善情况。如果我们只看这个特定的submaps，那么错误就只包含在这个submaps中。我们也看到，随着时间的推移，全局SLAN会发现发生这些奇怪的情形，并能部分的进行纠正，然而这些有问题的submaps就永远有问题了。 由于这儿的问题是在submaps中有滑动，这是局部SLAM的问题，所以我们可以先关掉全局SLAM,以免影响我们调优。

*POSE\_GRAPH.optimize\_every\_n\_nodes****=****0*

**submaps的正确尺寸**

随着时间的推移，只有闭环才能解决局部SLAM的这种漂移。submaps必须足够小，这样它们内部的偏移就会低于分辨率，因此它们在局部是正确的。另一方面，它应该足够大，，保证闭环能够正常工作。 submaps的尺寸通过TRAJECTORY\_BUILDER\_2D.submaps.num\_range\_data进行配置。通过查看这个示例的各个submaps，它们已经很好地适应了这两个约束条件，因此我们假设这个参数已经很好地进行了调整。

**扫描匹配器（ScanMatcher）的选择**

局部SLAM背后的理念是使用除了测距仪之外的传感器数据来预测下一个扫描应该被插入到submap的位置。然后，CeresScanMatcher将此作为先验，并找到扫描匹配到的submap的最佳位置。它通过插入submap 和子像素来对齐扫描。这是快速的，但是不能修复比submaps的分辨率大得多的错误。如果你的传感器设置和时间是合理的，只使用CeresScanMatcher通常是最好的选择。

如果你没有其他传感器或你不相信它们,Cartographer 还提供了一个RealTimeCorrelativeScanMatcher。它使用的方法类似于在闭环中对submap进行匹配，而不是与当前的submap相匹配。然后最好的匹配用作CeresScanMatcher的先验。这个扫描设备非常昂贵并且基本上不顾除测距仪外其他传感器的任何信号，但它在特征丰富的环境中是很稳定的。

**相关的扫描匹配器的调优**

待完善

**CeresScanMatcher的调优**

在我们的例程中，扫描匹配器可以在不影响匹配值的情况下，自由的向前或向后移动这个匹配。我们通过让扫描匹配器为之前获得的偏离付出更高的代价来避免这种情形，控制这个的两个参数是rotation\_weight和TRAJECTORY\_BUILDER\_2D.ceres\_scan\_matcher.translation\_weight。值越高，从之前的结果中移除它就需要越高的代价。换句话说：扫描匹配器在另一个位置必须生成一个更高的匹配值才能被接受。

出于教学的目的，让我们从先前的结果中制造一个相当大的偏离：

*TRAJECTORY\_BUILDER\_2D.ceres\_scan\_matcher.translation\_weight****=****1e3*

这使得优化器能够很好的充分的覆盖扫描结果，这一结果与之前的情况很接近，但与深度传感器不一致，而且明显是损坏的。对这个值进行试验会得到更好的结果，结果是2e2。

在这里，扫描匹配器使用旋转来稍微弄乱结果。将rotation\_weight设置为4e2，给我们带来了一个合理的结果。

### **2.3 验证**

为了确保我们没有对这个特定的问题进行过度优化，我们需要对其他收集到的数据进行配置。在这种情况下，例程b2-2016-04-05-14-44-52.bag的开始阶段，新的参数确实显示了下滑，所以我们不得不降低 translation\_weight 到1e2。对于我们想解决的问题，这种设置是的效果更差，但是不再下滑了，在检验它们之前，我们将所有的权重都标准化，因为它们只有相对意义，这个调优的结果在 [PR 428](https://github.com/googlecartographer/cartographer/pull/428)。一般来说，总是试着调优一个平台，而不是一个特定的包。

## **三、ROS API**

### **3.1 Cartographer 节点**

[Cartographer \_node](https://github.com/googlecartographer/cartographer_ros/blob/master/cartographer_ros/cartographer_ros/node_main.cc)用的是在线的实时的SLAM。

**命令行标志**

待确认(hrapp):这些不应该被删除吗？这看起来像是到处都在重复的记录。

–*configuration\_directory*

首先搜寻的配置文件的目录，然后是Cartographer 的安装目录允许包含的文件

–*configuration\_basename*

配置文件（如backpack\_3d.lua）的基本名称（也就是不包含任何目录前缀）

–*map\_filename*

从磁盘加载的Cartographer 状态文件。允许从之前的状态中增加新的SLAM轨迹，但是被加载的状态是被冻结的。

**Subscribed topics**

以下的范围数据topics是互斥的。至少需要一个范围数据源。

*scan (*[*sensor\_msgs/LaserScan*](http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html)*)*

支持二维和三维(例如，使用轴向旋转的平面激光扫描仪)。如果num\_laser\_scans在[Configuration](https://google-cartographer-ros.readthedocs.io/en/latest/configuration.html)配置为1，那么这个topic 将被用SLAM的输入。如果num\_laser\_scans大于1，那么多个被编号的scan topics (比如scan1、scan2、scan3、……直到并包括num\_laser\_scans)将被用作SLAM的输入。

*echoes (*[*sensor\_msgs/MultiEchoLaserScan*](http://docs.ros.org/api/sensor_msgs/html/msg/MultiEchoLaserScan.html)*)*

支持二维和三维(例如，使用轴向旋转的平面激光扫描仪)。如果 num\_multi\_echo\_laser\_scans在[Configuration](https://google-cartographer-ros.readthedocs.io/en/latest/configuration.html)配置为1，那么这个topic 将被用SLAM的输入。只使用第一个echo。如果num\_multi\_echo\_laser\_scans大于1，那么多个被编号的echoes topics (比如echoes\_1、echoes\_2、echoes\_3、……直到并包括num\_multi\_echo\_laser\_scans)将被用作SLAM的输入。

*points2 (*[*sensor\_msgs/PointCloud2*](http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html)*)*

如果 num\_point\_clouds在[Configuration](https://google-cartographer-ros.readthedocs.io/en/latest/configuration.html)配置为1，那么这个topic 将被用SLAM的输入。如果num\_point\_clouds大于1，那么多个被编号的points2 topics (比如points2\_1、points2\_2、points2\_3、……直到并包括num\_point\_clouds)将被用作SLAM的输入。

还可以提供以下附加的传感器数据topics。

*imu (*[*sensor\_msgs/Imu*](http://docs.ros.org/api/sensor_msgs/html/msg/Imu.html)*)*

支持2D(可选)和3D(必需)。这个topic 将被用作SLAM的输入。

*odom (*[*nav\_msgs/Odometry*](http://docs.ros.org/api/nav_msgs/html/msg/Odometry.html)*)*

支持2D(可选)和3D(可选)。如果在[Configuration](https://google-cartographer-ros.readthedocs.io/en/latest/configuration.html)中使能了use\_odometry，那么这个topic将被用作SLAM的输入。

### **Published topics**

### *scan\_matched\_points2 (sensor\_msgs/PointCloud2)*

### 用来进行 scan-to-submap匹配的点云，这个云可以根据配置进行滤波和投影。

*submap\_list (cartographer\_ros\_msgs/SubmapList)*

所有轨迹的所有submaps的列表，包含有没给submap的姿态和最新的版本号，

### **Services**

*submap\_query (cartographer\_ros\_msgs/SubmapQuery)*

获取所请求的submap。

*start\_trajectory (cartographer\_ros\_msgs/StartTrajectory)*

通过将传感器topics和轨迹选项指定为原始的二进制编码格式，开始另一个轨迹。返回一个已分配的轨迹ID。

*finish\_trajectory (cartographer\_ros\_msgs/FinishTrajectory)*

通过运行最终的优化，完成给定轨迹ID的轨迹。

*write\_state (cartographer\_ros\_msgs/WriteState)*

将当前的内部状态保存为指定的文件名并写入磁盘，这个文件通常以~/.ros结尾，若设置了ROS\_HOME，则以设置的ROS\_HOME结尾。该文件可以作为assets\_writer\_main的输入，用于生成诸如概率栅格、x射线或PLY文件等。

**需要的tf Transforms**

从所有传入的传感器数据帧转换到配置的tracking\_frame和published\_frame都必须是可用的。通常情况下，这些都是由 robot\_state\_publisher或static\_transform\_publisher定期发布的。

**提供的tf Transforms**

总是提供配置的map\_frame和published\_frame之间的转换。

如果在配置中使能provide\_odom\_frame，就会提供配置的odom\_frame和published\_frame 之间的连续(即不受闭环的影响)的转换。

### **3.2 离线节点**

[offline\_node](https://github.com/googlecartographer/cartographer_ros/blob/master/cartographer_ros/cartographer_ros/offline_node_main.cc)是使用传感器数据包进行SLAM最快速的方法。它不监听任何topics，而是从命令行中提供的一组包中读取TF和传感器数据。它还发布了一个带有先进传感器数据的时钟，也就是取代了rosbag play。在其他方面，它就像cartographer\_node一样。每个包最终都将成为一个单独的轨迹。一旦处理完所有的数据，它就会写出最终的Cartographer 状态并退出。

### **3.3 占据栅格节点**

[occupancy\_grid\_node](https://github.com/googlecartographer/cartographer_ros/blob/master/cartographer_ros/cartographer_ros/occupancy_grid_node_main.cc) 监听SLAM 发布的submaps，然后构建并发布ROS的occupancy\_grid。这个工具为了保持旧节点，需要一个单独的完全统一的地图，直到新的导航堆栈能够立即处理Cartographer’s submaps。生成这个地图比较慢，代价也比较大，所以地图更新是按秒进行排序的。

**Subscribed Topics**

只订阅Cartographer’s *submap\_list* topic。

**Published Topics**

*map (nav\_msgs/OccupancyGrid)*

如果被订阅，此节点将连续计算并发布地图。更新的时间随着地图尺寸的增加而增加，为了快速的更新，要使用submaps APIs。

|  |  |
| --- | --- |
| include "map\_builder.lua" |  |
|  | include "trajectory\_builder.lua" |
|  |  |
|  | options = { |
|  | map\_builder = MAP\_BUILDER, |
|  | trajectory\_builder = TRAJECTORY\_BUILDER, |
|  | map\_frame = "map", |
|  | tracking\_frame = "imu\_link", |
|  | published\_frame = "odom", |
|  | odom\_frame = "odom", |
|  | provide\_odom\_frame = false, |
|  | publish\_frame\_projected\_to\_2d = false, |
|  | use\_odometry = true, |
|  | use\_nav\_sat = false, |
|  | use\_landmarks = false, |
|  | num\_laser\_scans = 1, |
|  | num\_multi\_echo\_laser\_scans = 0, |
|  | num\_subdivisions\_per\_laser\_scan = 1, |
|  | num\_point\_clouds = 0, |
|  | lookup\_transform\_timeout\_sec = 0.2, |
|  | submap\_publish\_period\_sec = 0.3, |
|  | pose\_publish\_period\_sec = 5e-3, |
|  | trajectory\_publish\_period\_sec = 30e-3, |
|  | rangefinder\_sampling\_ratio = 1., |
|  | odometry\_sampling\_ratio = 1., |
|  | fixed\_frame\_pose\_sampling\_ratio = 1., |
|  | imu\_sampling\_ratio = 1., |
|  | landmarks\_sampling\_ratio = 1., |
|  | } |
|  |  |
|  | MAP\_BUILDER.use\_trajectory\_builder\_2d = true |
|  |  |
|  | TRAJECTORY\_BUILDER\_2D.min\_range = 0.1 |
|  | TRAJECTORY\_BUILDER\_2D.max\_range = 8. |
|  | TRAJECTORY\_BUILDER\_2D.missing\_data\_ray\_length = 5. |
|  | TRAJECTORY\_BUILDER\_2D.use\_imu\_data = true |
|  | TRAJECTORY\_BUILDER\_2D.use\_online\_correlative\_scan\_matching = true |
|  | TRAJECTORY\_BUILDER\_2D.motion\_filter.max\_angle\_radians = math.rad(0.1) |
|  |  |
|  | POSE\_GRAPH.constraint\_builder.min\_score = 0.65 |
|  | POSE\_GRAPH.constraint\_builder.global\_localization\_min\_score = 0.7 |
|  |  |
|  | return options |