6、雷达建图

Gmapping：http://wiki.ros.org/gmapping/

hector_slam：http://wiki.ros.org/hector_slam

hector_slam/Tutorials：http://wiki.ros.org/hector_slam/Tutorials/SettingUpForYourRobot

hector_mapping：http://wiki.ros.org/hector_mapping

karto：http://wiki.ros.org/slam_karto

Cartographer：https://google-cartographer.readthedocs.io/en/latest/

Cartographer ROS：https://google-cartographer-ros.readthedocs.io/en/latest/

rrt_exploration：http://wiki.ros.org/rrt_exploration

rrt_exploration/Tutorials：http://wiki.ros.org/rrt_exploration/Tutorials

map_server：https://wiki.ros.org/map_server

6.1、使用方法

6.1.1、建图

注意：建图时，速度越慢效果越好（注要是旋转速度要慢些），速度太快，效果会很差。

启动命令（注意：纯深度建图导航效果不佳，不推荐使用）。

```
roslaunch transbot_nav usbcam_bringup.launch lidar_type:=a1   # mono + laser +
Transbot
roslaunch transbot_nav astra_bringup.launch                   # Astra + Transbot
roslaunch transbot_nav laser_bringup.launch lidar_type:=a1    # laser + Transbot
roslaunch transbot_nav transbot_bringup.launch lidar_type:=a1 # Astra + laser +
Transbot
```

lidar_type参数：使用激光雷达的型号：[a1,a2,a3,s1,s2]。

建图

```
roslaunch transbot_nav transbot_map.launch map_type:=gmapping
roslaunch transbot_nav transbot_map.launch map_type:=hector
roslaunch transbot_nav transbot_map.launch map_type:=karto
roslaunch transbot_nav transbot_map.launch map_type:=cartographer
roslaunch transbot_nav rrt_exploration.launch
```

- 参数【map_type】：建图算法【gmapping,hector,karto,cartographer】，默认是【gmapping】。

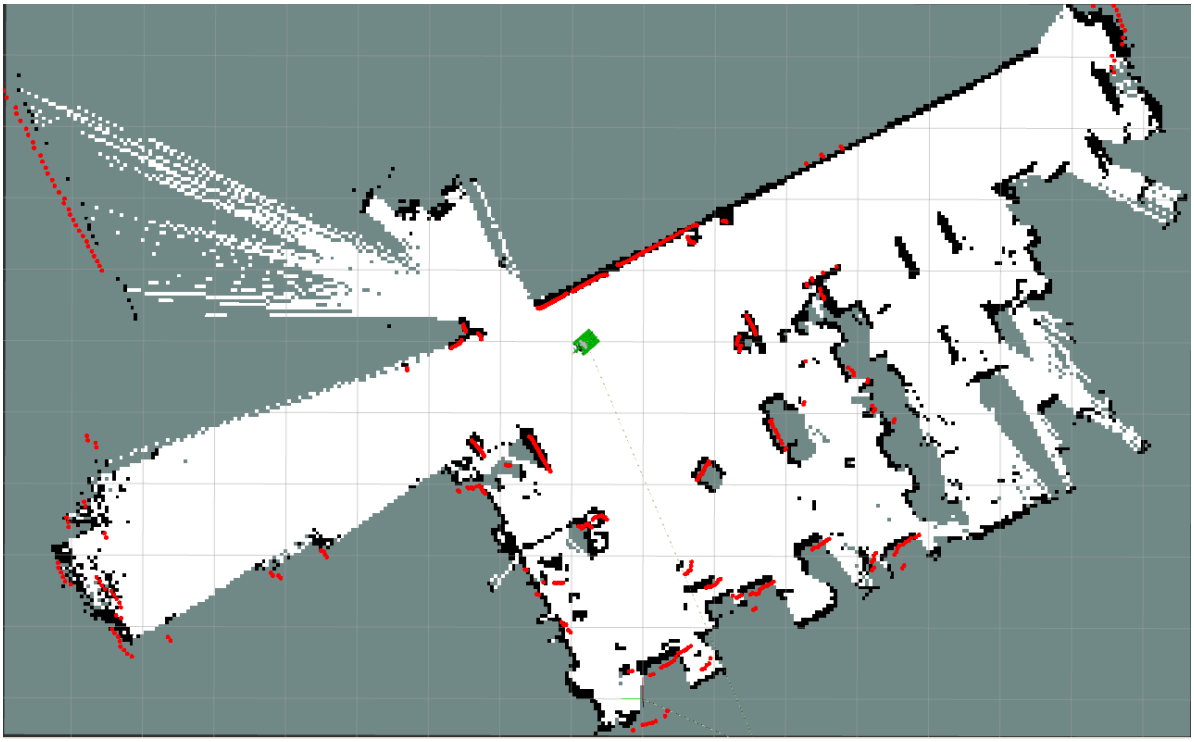
6.1.2、使用

键盘控制机器人移动

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

手柄控制机器人移动

使得机器人走满要建图的区域，地图尽可能闭合。



建图的过程中可能会有一些散射出去的点。如果建图环境闭合很好，比较规则，运动较慢，散射现象就小很多。

6.1.3、地图保存

几种建图算法保存地图的方式是不同的，

- cartographer：执行下列命令

```
bash ~/rplidar_ws/src/transbot_nav/maps/carto_map.sh
```

- rrt_exploration：按照要求选好五个点后，机器人开始探索建图，见图完毕，自动保存地图，返回零点。地图被保存到~/rplidar_ws/src/transbot_nav/maps/my_map文件夹下，名称是rrt_map，可在launch文件中修改。
- gmapping,hector,karto：执行下面命令保存即可。

```
rosrun map_server map_saver -f ~/rplidar_ws/src/transbot_nav/maps/my_map    # 第一种方式
bash ~/rplidar_ws/src/transbot_nav/maps/map.sh                              # 第二种方式
```

地图将被保存到~/rplidar_ws/src/transbot_nav/maps/文件夹下，一个pgm图片，一个yaml文件。

map.yaml

```
image: map.pgm
resolution: 0.05
origin: [-15.4,-12.2,0.0]
negate: 0
occupied_thresh: 0.65
free_thresh:  0.196
```

参数解析：

- image：地图文件的路径，可以是绝对路径，也可以是相对路径

- resolution：地图的分辨率， 米/像素
- origin：地图左下角的 2D 位姿(x,y,yaw), 这里的yaw是逆时针方向旋转的（yaw=0 表示没有旋转）。目前系统中的很多部分会忽略yaw值。
- negate：是否颠倒 白/黑 、自由/占用 的意义（阈值的解释不受影响）
- occupied_thresh：占用概率大于这个阈值的的像素，会被认为是完全占用。
- free_thresh：占用概率小于这个阈值的的像素，会被认为是完全自由。

6.1.4、查看相关信息

查看tf树

```
rosrun rqt_tf_tree rqt_tf_tree
```

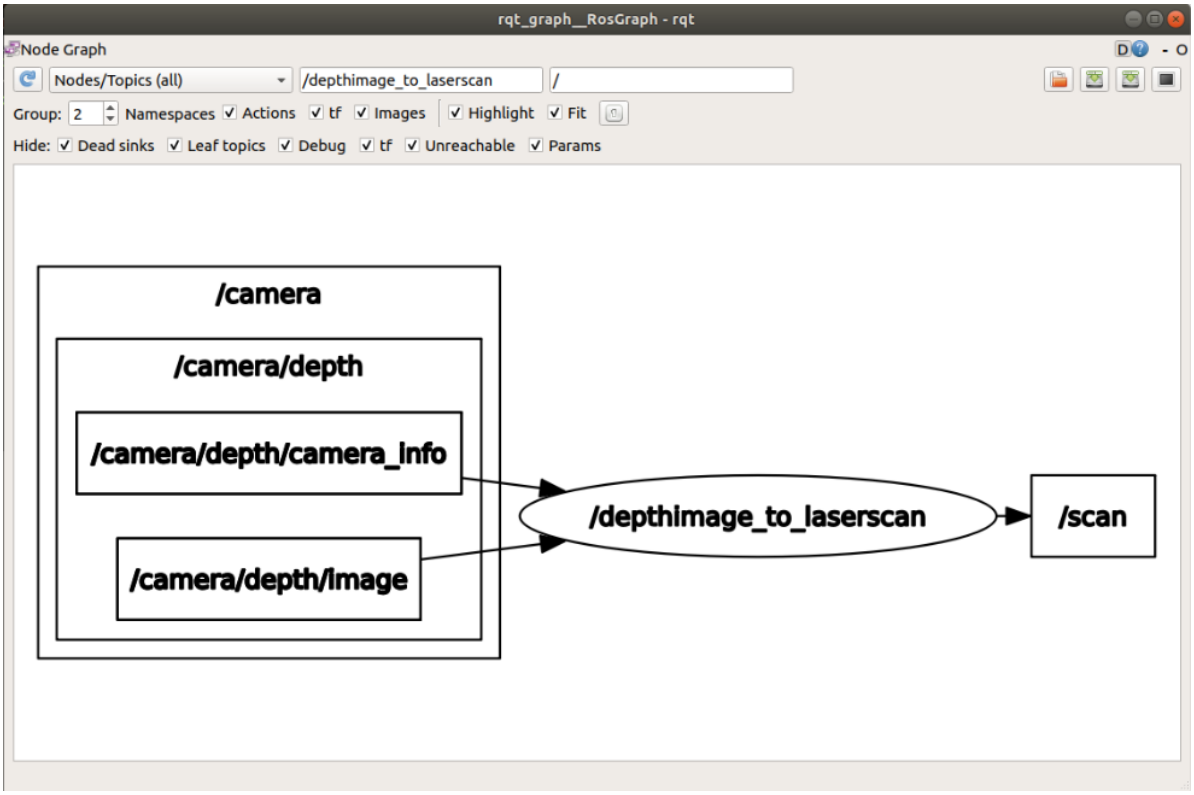
节点查看

```
rqt_graph
```

6.1.5、纯视觉2D建图

启动驱动参考【6.1.1】# Astra + Transbot；启动建图命令和【6.1.1】一样。主要使用功能包depthimage_to_laserscan，将深度图像转化为激光雷达数据，其建图功能和激光雷达相同。注意：深度相机的扫描范围不是360°。

```
rqt_graph
```



6.2、建图算法

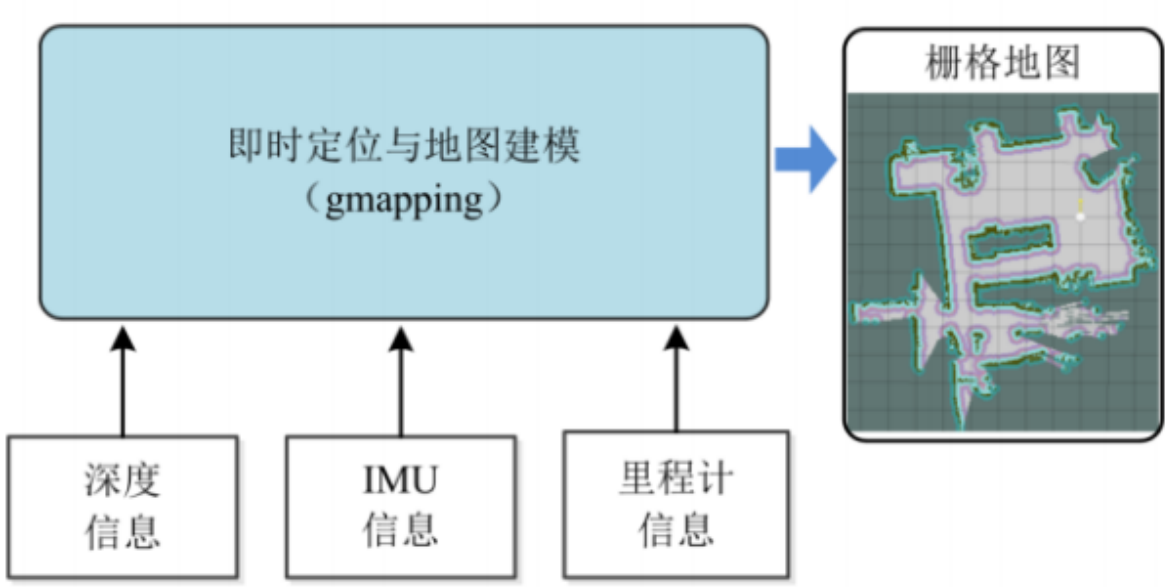
6.2.1、gmapping

1）简介

- Gmapping是基于滤波SLAM框架的常用开源SLAM算法。
- Gmapping基于RBpf粒子滤波算法，即将定位和建图过程分离，先进行定位再进行建图。
- Gmapping在RBpf算法上做了两个主要的改进：改进提议分布和选择性重采样。

优点：Gmapping可以实时构建室内地图，在构建小场景地图所需的计算量较小且精度较高。

缺点：随着场景增大所需的粒子增加，因为每个粒子都携带一幅地图，因此在构建大地图时所需内存和计算量都会增加。因此不适合构建大场景地图。并且没有回环检测，因此在回环闭合时可能会造成地图错位，虽然增加粒子数目可以使地图闭合但是以增加计算量和内存为代价。



2）话题和服务

订阅话题	**类型**	**描述**
tf	tf/tfMessage	用于激光雷达坐标系、基坐标系、里程计坐标系之间转换
scan	sensor_msgs/LaserScan	激光雷达扫描数据
发布话题	**类型**	**描述**
map_metadata	nav_msgs/MapMetaData	发布地图Meta数据
map	nav_msgs/OccupancyGrid	发布地图栅格数据
~entropy	std_msgs/Float64	发布机器人姿态分布熵的估计
服务	**类型**	**描述**
dynamic_map	nav_msgs/GetMap	获取地图数据

3）配置参数

参数	类型	默认值	描述
~throttle_scans	int	1	每次接收到该数量帧的激光数据后只处理其中的一帧数据，默认每接收到一帧数据就处理
~base_frame	string	"base_link"	机器人基坐标系
~map_frame	string	"map"	地图坐标系
~odom_frame	string	"odom"	里程计坐标系
~map_update_interval	float	5.0	地图更新频率，该值越低，计算负载越大
~maxUrange	float	80.0	激光可探测的最大范围
~sigma	float	0.05	端点匹配的标准差
~kernelSize	int	1	在对应的内核中进行查找
~lstep	float	0.05	平移过程中的优化步长
~astep	float	0.05	旋转过程中的优化步长
~iterations	int	5	扫描匹配的迭代次数
~lsigma	float	0.075	似然计算的激光标准差
~ogain	float	3.0	似然计算时用于平滑重采样效果
~lskip	int	0	每次扫描跳过的光束数
~minimumScore	float	0	扫描匹配结果的最低值。当使用有限范围（例如5m）的激光扫描仪时，可以避免在大开放空间中跳跃姿势估计
~srr	float	0.1	平移函数（rho/rho），平移时的里程计误差
~srt	float	0.2	旋转函数（rho/theta），平移时的里程计误差
~str	float	0.1	平移函数（theta/rho），旋转时的里程计误差
~stt	float	0.2	旋转函数（theta/theta），旋转时的里程计误差
~linearUpdate	float	1.0	机器人每平移该距离后处理一次激光扫描数据
~angularUpdate	float	0.5	机器人每旋转该距离后处理一次激光扫描数据
~temporalUpdate	float	-1.0	如果最新扫描处理的速度比更新的速度慢，则处理一次扫描。该值为负数时关闭基于时间的更新

参数	类型	默认值	描述
~resampleThreshold	float	0.5	基于Neff的重采样阈值
~particles	int	30	滤波器的粒子数目
~xmin	float	-100.0	地图x向初始最小尺寸
~ymin	float	-100.0	地图y向初始最小尺寸
~xmax	float	100.0	地图x向初始最大尺寸
~ymax	float	100.0	地图y向初始最大尺寸
~delta	float	0.05	地图分辨率
~llsamplerange	float	0.01	似然计算的平移采样距离
~llsamplestep	float	0.01	似然计算的平移采样步长
~lasamplerange	float	0.005	似然计算的旋转采样距离
~lasamplestep	float	0.005	似然计算的旋转采样步长
~transform_publish_period	float	0.05	TF变换发布的时间间隔
~occ_threh	float	0.25	栅格地图占用率的阈值
~maxRange(float)	float	-	传感器的最大范围

4）TF变换

必需的TF变换	描述
laser-->base_link	激光雷达坐标系与基坐标系之间的变换，一般由robot_state_publisher或者static_transform_publisher发布
base_link-->odom	地图坐标系与机器人里程计坐标系之间的变换，估计机器人在地图中的位姿
发布的TF变换	**描述**
map-->odom	地图坐标系与机器人里程计坐标系之间的变换，估计机器人在地图中的位姿

6.2.2、hector

1）简介

特点：hector_slam不需要订阅里程计/odom消息，使用高斯牛顿方法，直接使用激光雷达估算里程计信息。但是机器人速度较快时会发生打滑现象，导致建图效果出现偏差，对传感器的要求高。在建图的时候，尽可能将小车旋转速度调至较低。

没有odom坐标系使用方法，摘自Wiki。

2. Use without odom frame

If you do not require the use of a odom frame (for example because your platform does not provide any usable odometry) you can directly publish a transformation from map to base_link:

```
<param name="pub_map_odom_transform" value="true"/>
<param name="map_frame" value="map" />
<param name="base_frame" value="base_frame" />
<param name="odom_frame" value="base_frame" />
```

2）话题和服务

话题订阅	**类型**	**描述**
scan	sensor_msgs/LaserScan	激光雷达扫描的深度数据
syscommand	std_msgs/String	系统命令。如果字符串等于“reset”，则地图和机器人姿态重置为初始状态
话题发布	**类型**	**描述**
map_metadata	nav_msgs/MapMetaData	发布地图Meta数据
map	nav_msgs/OccupancyGrid	发布地图栅格数据
slam_out_pose	geometry_msgs/PoseStamped	无协方差的机器人姿态估计
poseupdate	geometry_msgs /PoseWithCovarianceStamped	具有高斯不确定性估计的机器人姿态估计
服务	**类型**	**描述**
dynamic_map	nav_msgs/GetMap	获取地图数据
reset_map	std_srvs/Trigger	调用此服务重置地图，hector将从头开始创建一个全新的地图。请注意，这不会重新启动机器人的姿势，它将从上次录制的姿势重新启动。
pause_mapping	std_srvs/SetBool	调用此服务以停止/开始处理激光扫描。
restart_mapping_with_new_pose	hector_mapping/ResetMapping	调用此服务重置地图、机器人的姿势，并恢复地图（如果暂停）

3）配置参数

参数	类型	默认值	描述
~base_frame	String	"base_link"	机器人基坐标系，用于定位和激光扫描数据的变换
~map_frame	String	"map"	地图的坐标系
~odom_frame	string	"odom"	里程计坐标系（实质是map指向的坐标系）
~map_resolution	Double	0.025(m)	地图分辨率，网格单元的边缘长度
~map_size	Int	1024	地图的大小
~map_start_x	double	0.5	/map的原点【0.0，1.0】在x轴上相对网格图的位置
~map_start_y	double	0.5	/map的原点【0.0，1.0】在y轴上相对网格图的位置
~map_update_distance_thresh	double	0.4(m)	地图更新的阈值，在地图上从一次更新起算到直行距离到达该参数值后再次更新
~map_update_angle_thresh	double	0.9(rad)	地图更新的阈值，在地图上从一次更新起算到旋转达到达该参数值后再次更新
~map_pub_period	double	2.0	地图发布周期
~map_multi_res_levels	int	3	地图多分辨率网格级数
~update_factor_free	double	0.4	用于更新空闲单元的地图，范围是【0.0，1.0】
~update_factor_occupied	double	0.9	用于更新被占用单元的地图，范围是【0.0，1.0】
~laser_min_dist	double	0.4(m)	激光扫描点的最小距离，小于此值的扫描点将被忽略
~laser_max_dist	double	30.0(m)	激光扫描点的最大距离，小于此值的扫描点将被忽略

参数	类型	默认值	描述
~laser_z_min_value	double	-1.0(m)	相对于激光雷达的最小高度，低于此值的扫描点将被忽略
~laser_z_max_value	double	1.0(m)	相对于激光雷达的最大高度，高于此值的扫描点将被忽略
~pub_map_odom_transform	bool	true	是否发布map与odom之间的坐标转换
~output_timing	bool	false	通过ROS_INFO处理每个激光扫描的输出时序信息
~scan_subscrible_queue_size	int	5	扫描订阅者的队列大小
~pub_map_scanmatch_transform	bool	true	是否发布scanmatcher与map之间的坐标变换
~tf_map_scanmatch_transform_frame_name	String	"scanmatcher_frame"	scanmatcher的坐标系命名

4）TF变换

必需的TF变换	描述
laser-->base_link	通常为固定值，激光雷达坐标系与基坐标系之间的变换，一般由robot_state_publisher或者static_transform_publisher发布
发布的TF变换	**描述**
map-->odom	地图框内机器人姿势的当前估计值（仅当参数“pub_map_odom_transform”为真时提供）。

6.2.3、karto

1）简介

Karto是一种2D激光SLAM解决方案，它是基于稀疏图优化的方法，带闭环检测。karto采取的是spa(karto_slam)或g2o(nav2d)优化库，前端与后端采取的是单线程进行。利用的是odom进行初始位置的预测

2）话题和服务

话题订阅	类型	描述
scan	sensor_msgs/LaserScan	激光雷达扫描的深度数据
tf	tf/tfMessage	用于激光雷达坐标系、基坐标系、里程计坐标系之间转换
话题发布	**类型**	**描述**
map_metadata	nav_msgs/MapMetaData	发布地图Meta数据
map	nav_msgs/OccupancyGrid	发布地图栅格数据
visualization_marker_array	visualisation_msgs / MarkerArray	发布姿势图
服务	**类型**	**描述**
dynamic_map	nav_msgs/GetMap	获取地图数据

3）配置参数

- 通用参数

参数	类型	默认值	说明
~base_frame	string	"base_link"	机器人基坐标系
~map_frame	string	"map"	地图坐标系
~odom_frame	string	"odom"	里程计坐标系
~throttle_scans	int	1	每次这么多扫描中处理1次（将其设置为更高的数字以跳过更多扫描）
~map_update_interval	float	5.0	地图更新之间的间隔时间（秒）。降低此数字会更频繁地更新占用率网格，但会增加计算负载。
~resolution	float	0.05	地图分辨率（每个占用网格块的米数）
~delta	float	0.05	地图分辨率（每个占用网格块的米数）。与resolution相同。为与gmapping的参数名兼容而定义。
~transform_publish_period	float	0.05	转换发布之间的间隔时间（秒）。
use_scan_matching	bool	true	是否使用扫描匹配算法，一般情况下设置为true，mapper算法可以修正里程计与激光的噪声和误差。在一些传感器数据精准的仿真环境中，扫描匹配算法反倒会获得更差的效果（因为使用了高斯模糊，反而降低了高精度传感器的观测置信度），建议关闭（仿真环境加入噪声即可）。
use_scan_barycenter	bool	true	使用扫描端点的质心定义扫描之间的距离。
minimum_travel_distance	double	0.2	设置扫描之间的最小行程。
minimum_travel_heading	double	deg2rad(10)=0.087266461	设置扫描之间的最小角度。
scan_buffer_size	int	70	设置 ScanBuffer 的长度，约等于 scan_buffer_maximum_scan_distance/minimum_travel_distance
scan_buffer_maximum_scan_distance	double	20.0	设置 ScanBuffer 的最大长度和 Size作用类似
link_match_minimum_response_fine	double	0.8	设置最小scans 连接 的最小响应阈值
link_scan_minimum_distance	double	10.0	设置两个连接的 scans 最大距离， 大于此值则不考虑两者的响应阈值
loop_search_maximum_distance	double	4.0	回环检测最大距离。距离当前位置小于此距离的扫描将被视为匹配环路闭合。
do_loop_closing	bool	true	是否启用回环检测
loop_match_minimum_chain_size	int	10	回环检测最下链条尺寸
loop_match_maximum_variance_coarse	double	math::Square(0.4)=0.16	回环匹配时粗匹配的最大协方差值， 小于此值才认为是一个可行解
loop_match_minimum_response_coarse	double	0.8	回环匹配时粗匹配的最小响应， 响应值大于此值将会开始粗精度的回环优化
loop_match_minimum_response_fine	double	0.8	回环匹配最小响应阈值，大于此值才开始进行高精度

- 矫正参数

参数	类型	默认值	说明
correlation_search_space_dimension	double	0.3	设置Correlation Grid的搜索范围大小
correlation_search_space_resolution	double	0.01	设置Correlation Grid的分辨率
correlation_search_space_smear_deviation	double	0.03	设置Correlation Grid模糊程度

- 回环参数

参数	类型	默认值	说明
loop_search_space_dimension	double	8.0	回环检测空间范围大小
loop_search_space_resolution	double	0.05	回环检测空间分辨率
loop_search_space_smear_deviation	double	0.03	回环检测模糊程度

- Scan Matcher参数

参数	类型	默认值	说明
distance_variance_penalty	double	sqrt(0.3)=0.09(要小于1.0)	scan-matching时 对里程计的补偿系数
angle_variance_penalty	double	sqrt(deg2rad(20))=0.17453292	scan-matching时 对角度的补偿系数
fine_search_angle_offset	double	deg2rad(0.2)=0.0017453292	精搜索角度范围
coarse_search_angle_offset	double	deg2rad(20)=0.17453292	粗搜索角度范围
coarse_angle_resolution	double	deg2rad(2)=0.017453292	粗搜索角分辨率
minimum_angle_penalty	double	0.9	最小角度惩罚
minimum_distance_penalty	double	0.5	最下距离惩罚
use_response_expansion	bool	false	在没有发现好的匹配的情况下，是否增加搜索范围

4）TF变换

必需的TF变换	描述
laser-->base_link	通常为固定值，激光雷达坐标系与基坐标系之间的变换，一般由robot_state_publisher或者static_transform_publisher发布
base_link-->odom	地图坐标系与机器人里程计坐标系之间的变换，估计机器人在地图中的位姿
发布的TF变换	**描述**
map-->odom	地图框内机器人姿势的当前估计值（仅当参数“pub_map_odom_transform”为真时提供）。

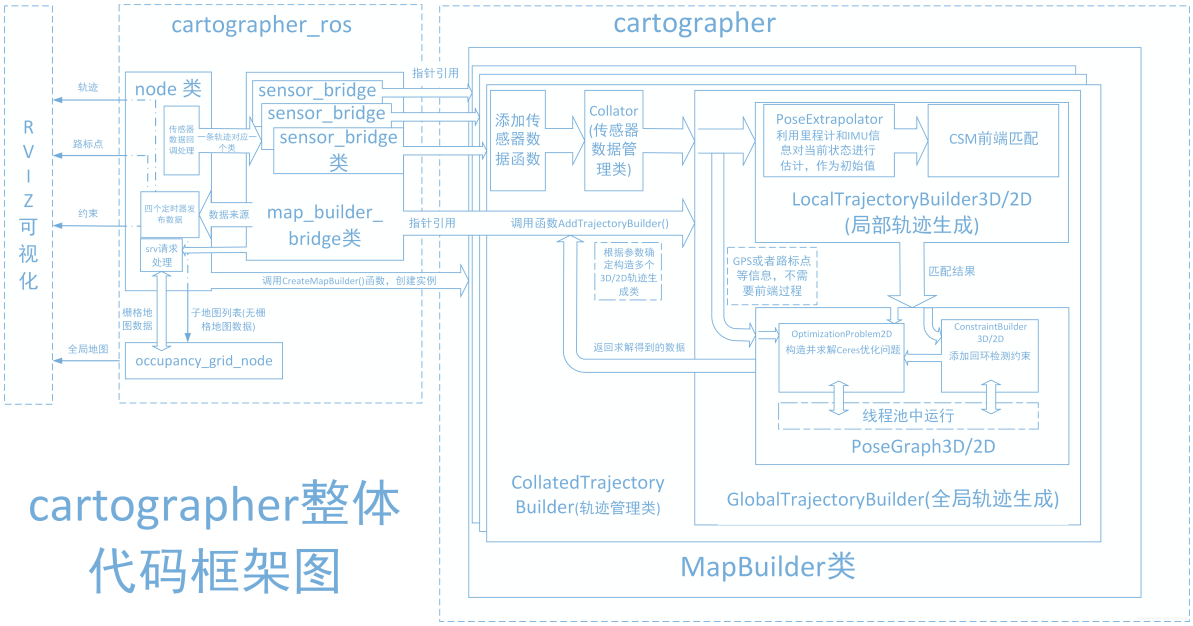
6.2.4、cartographer

1）简介

Cartographer，是Google开源的一个ROS系统支持的2D和3D SLAM（simultaneous localization and mapping）库。基于图优化（多线程后端优化、cere构建的problem优化）的方法建图算法。可以结合来自多个传感器（比如，LIDAR、IMU 和 摄像头）的数据，同步计算传感器的位置并绘制传感器周围的环境。

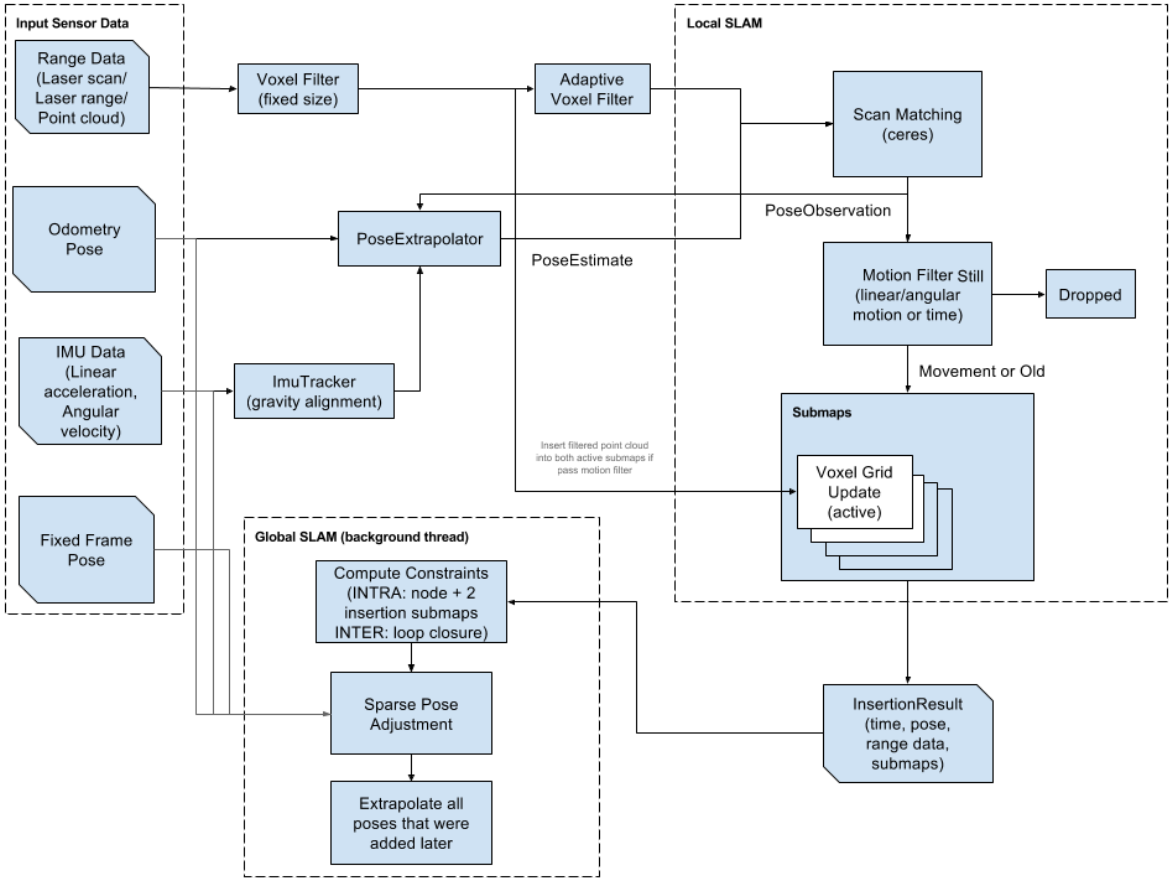
2）代码结构

cartographer的源码主要包括三个部分：cartographer、cartographer_ros和ceres-solver（后端优化）。



cartographer整体代码框架图

- cartographer



cartographer采用的是主流的SLAM框架，也就是特征提取、闭环检测、后端优化的三段式。由一定数量的LaserScan组成一个submap子图，一系列的submap子图构成了全局地图。用LaserScan构建submap的短时间过程累计误差不大，但是用submap构建全局地图的长时间过程就会存在很大的累计误差，所以需要利用闭环检测来修正这些submap的位置，闭环检测的基本单元是submap，闭环检测采用scan_match策略。cartographer的重点内容就是融合多传感器数据（odometry、IMU、LaserScan等）的submap子图创建以及用于闭环检测的scan_match策略的实现。

- cartographer_ros

cartographer_ros这个package是在ROS下面运行的，可以以ROS消息的方式接受各种传感器数据，在处理过后又以消息的形式publish出去，便于调试和可视化。

3）配置参数

lua文件

参数	说明
map_frame	地图坐标系
tracking_frame	将所有传感器数据转换到这个坐标系下
published_frame	map指向的坐标系
odom_frame	是否提供odom的tf？如果为true，则tf树为map->odom->footprint；如果为false，tf树为map->footprint
provide_odom_frame	如果为true，the local, non-loop-closed, continuous pose将会在map_frame里以odom_frame发布?
publish_frame_projected_to_2d	如果启用，则已发布的pose将限制为纯2D姿势（无滚动，俯仰或z偏移）
use_odometry	是否使用里程计,如果使用要求一定要有odom的tf
use_nav_sat	是否使用gps
use_landmarks	是否使用landmark
num_laser_scans	是否使用单线激光数据
num_multi_echo_laser_scans	是否使用multi_echo_laser_scans数据
num_subdivisions_per_laser_scan	1帧数据被分成几次处理,一般为1
num_point_clouds	是否使用点云数据
lookup_transform_timeout_sec	查找tf时的超时时间
submap_publish_period_sec	发布submap的时间间隔（秒）
pose_publish_period_sec	发布pose的时间间隔，值为5e-3的时候为200HZ
trajectory_publish_period_sec	发布trajectory markers（trajectory的节点）的时间间隔，值为30e-3为30ms
rangefinder_sampling_ratio	激光雷达消息的固定采样频率
odometry_sampling_ratio	里程计消息的固定采样频率
fixed_frame_pose_sampling_ratio	固定坐标系消息的固定采样频率
imu_sampling_ratio	IMU 消息的固定采样频率
landmarks_sampling_ratio	路标消息的固定采样频率

6.3、rrt_exploration

6.3.1、简介

RRT exploration是基于RRT路径规划算法实现的搜索算法。之所以使用RRT算法是因为RRT对于未知区域有着强烈的倾向，在RRT exploration中，RRT主要用于生成边界点，这样对于探索边界点是很有好处的。所谓的边界点就是已经探索过的和未知的区域的交界点，在这里做一个定义:对于所有的区域，如果是探索过的，没有障碍物的区域记为0，有障碍物的记为1，未知的区域记为-1，开始时，整个区域都是记为-1的。

快速探索随机树 (RRT) 算法的框架如下:

- Global RRT frontier point detector node.
- Local RRT frontier point detector node.
- OpenCV-based frontier detector node.
- Filter node.
- Assigner node.

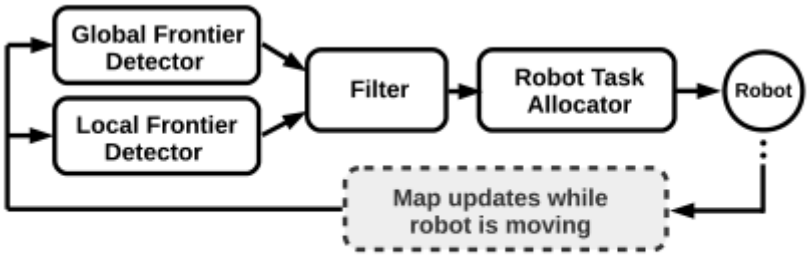


Fig. 1: Overall schematic diagram of the exploration algorithm

有3种类型的节点：用于检测占用栅格地图中的边界点的节点，用于过滤检测到的点的节点，以及用于将点分配给机器人的节点。

6.3.2、global_rrt_frontier_detector

1）简介

global_rrt_frontier_detector节点采用占用网格并在其中查找边界点（即勘探目标）。它发布检测到的点，以便过滤器节点可以进行处理。在多机器人配置中，仅运行该节点的一个实例。如果需要，运行全局边界检测器的其他实例可以提高边界点检测的速度。

2）话题和服务

订阅话题	类型	描述
map	nav_msgs/OccupancyGrid	话题名称由~map_Topic参数定义。它是该节点接收话题名称。
clicked_point	geometry_msgs/PointStamped	global_rrt_frontier_detector 节点要探测的区域。本话题是节点接收定义区域的五个点的位置。前四个点是定义要探索的正方形区域的四个点，最后一个点是树的起点。在公布这五点后，RRT将开始探测边境点。
发布话题	**类型**	**描述**
detected_points	geometry_msgs/PointStamped	发布检测到的边界点的话题。
shapes	visualization_msgs/Marker	发布RRT假想的线型以使用Rviz查看。

3）配置参数

参数	类型	默认值	描述
~map_topic	string	"/robot_1/map"	该节点接收地图话题映射名称
~eta	float	5.0	此参数控制用于检测边界点的RRT的增长率，单位为米。此参数应根据地图大小进行设置，非常大的值将导致树增长更快，从而更快地检测边界点，但较大的增长率也意味着树将丢失地图中的小角点

6.3.3、local_rrt_frontier_detector

1）简介

此节点类似于global_rrt_frontier_detector。但是，它的工作方式不同，因为每次检测到边界点时，这里的树都会不断重置。该节点将沿着全局边界检测器节点运行，它负责快速检测位于机器人附近的边界点。

在多机器人配置中，每个机器人运行一个本地探测器实例。因此，对于一个由3个机器人组成的团队，将有4个节点用于检测边界点；3个本地探测器和1个全局探测器。如果需要，运行本地边界检测器的其他实例可以提高边界点检测的速度。所有探测器将在同一话题上发布探测到的边界点（“/detected_points”）。

2）话题和服务

订阅话题	类型	描述
map	nav_msgs/OccupancyGrid	该节点订阅的地图话题名称。
clicked_point	geometry_msgs/PointStamped	与global_rrt_frontier_detector 节点类似
发布话题	**类型**	**描述**
detected_points	geometry_msgs/PointStamped	发布检测到的边界点的话题。
shapes	visualization_msgs/Marker	发布RRT假想的线型以使用Rviz查看。

3）配置参数

参数	类型	默认值	描述
~/robot_1/base_link	string	"/robot_1/base_link"	连接到机器人的框架。每次重置RRT树时，它将从这个坐标系中获取的当前机器人位置开始。
~map_topic	string	"/robot_1/map"	该节点接收地图话题映射名称
~eta	float	5.0	此参数控制用于检测边界点的RRT的增长率，单位为米。此参数应根据地图大小进行设置，非常大的值将导致树增长更快，从而更快地检测边界点，但较大的增长率也意味着树将丢失地图中的小角点

6.3.4、frontier_opencv_detector

1）简介

此节点是另一个边界检测器，但它不基于RRT。此节点使用OpenCV工具检测边界点。它旨在单独运行，在多机器人配置中，只应运行一个实例（运行此节点的其他实例不会产生任何差异）。

最初，实现该节点是为了与基于RRT的边界检测器进行比较。沿着RRT检测器（本地和全局）运行此节点可以提高边界点检测的速度。

注意：您可以运行任何类型和数量的检测器，所有检测器都将发布在过滤器节点（将在下一节中解释）订阅的同一主题上。另一方面，过滤器将过滤后的边界点传递给分配者，以便命令机器人探索这些点。

2）话题和服务

订阅话题	**类型**	**描述**
map	nav_msgs/OccupancyGrid	该节点订阅的地图话题名称。
发布话题	**类型**	**描述**
detected_points	geometry_msgs/PointStamped	发布检测到的边界点的话题。
shapes	visualization_msgs/Marker	发布RRT假想的线型以使用Rviz查看。

3）配置参数

参数	**类型**	**默认值**	**描述**
~map_topic	string	"/robot_1/map"	该节点接收地图话题映射名称

6.3.5、filter

1）简介

该节点节点从所有检测器接收检测到的边界点，过滤这些点，并将它们传递给分配节点以命令机器人。过滤包括删除旧点和无效点，还包括删除冗余点。

2）话题和服务

订阅话题	**类型**	**描述**
map	nav_msgs/OccupancyGrid	话题名称由~map_Topic参数定义。它是该节点接收话题名称。
robot_x/move_base_node/global_costmap/costmap	nav_msgs/OccupancyGrid	x是机器人的编号。该节点订阅所有机器人的所有代价地图的话题，因此需要costmap。通常情况下，代价地图应由navigation发布（在机器人上打开navigation后，每个机器人将有一个代价地图）。例如，如果n_robots=2，则节点将订阅：robot_1/move_base_node/global_costmap/costmap和robot_2/move_base_node/global_costmap/costmap。代价地图用于删除无效的点。 注意：与机器人对应的所有节点的名称空间应以robot_x开头。
detected_points	geometry_msgs/PointStamped	~goals_topic定义的话题名。它是过滤器节点接收边界检测点的话题。
发布话题	**类型**	**描述**
frontiers	visualization_msgs/Marker	过滤器节点仅发布过滤后的边界点的话题。
centroids	visualization_msgs/Marker	过滤器节点发布接收到的边界点的话题。
filtered_points	MsgLink(msg/type)	所有过滤后的点作为点数组发送到此主题的assigner节点。

3）配置参数

参数	类型	默认值	描述
~map_topic	string	"/robot_1/map"	该节点接收地图话题映射名称
~costmap_clearing_threshold	float	70.0	代价地图清理阈值
~info_radius	float	1.0	用于计算边界点信息增益的信息半径。
~goals_topic	string	/detected_points	定义节点接收检测边界点的主题
~n_robots	float	1.0	机器人数量
~namespace	string		命名空间
~namespace_init_count	float	1.0	命名空间的索引
~rate	float	100.0	节点循环速率（以Hz为单位）。

6.3.6、Assigner

1）简介

该节点接收目标探测目标，即过滤节点发布的过滤边界点，并相应地命令机器人。赋值器节点通过move_base_node命令机器人。这就是为什么要在机器人上启动导航。

2）话题和服务

订阅话题	类型	描述
map	nav_msgs/OccupancyGrid	话题名称由~map_Topic参数定义。它是该节点接收话题名称。
frontiers_topic	nav_msgs/OccupancyGrid	该话题名由 ~frontiers_topic 参数定义

3）配置参数

参数	类型	默认值	描述
~map_topic	string	"/robot_1/map"	该节点接收地图话题映射名称
~info_radius	float	1.0	用于计算边界点信息增益的信息半径。
~info_multiplier	float	3.0	单位是米。此参数用于强调边界点的信息增益相对于成本（到边界点的预期行程距离）的重要性。
~hysteresis_radius	float	3.0	单位是米。此参数定义滞后半径。
~hysteresis_gain	float	2.0	单位是米。此参数定义滞后增益。
~frontiers_topic	string	/filtered_points	该节点接收边界点的话题。
~n_robots	float	1.0	机器人数量
~namespace	string		命名空间
~namespace_init_count	float	1.0	正在启动机器人名称的索引。
~delay_after_assignement	float	0.5	单位是秒。它定义了每次机器人分配后的延迟量。
~global_frame	string	"/map"	用于全局坐标系。在单机器人中，它与“map_topic”参数相同。在多机器人的情况下，该坐标系名称相对应的是全局坐标系名称。