Cartographer-0-运行 Demo 数据集和 MIT 数据集

2020-03-16 08:48:37

Contents

1. 运行官方 Demo

德意志博物馆数据集

1.1. 下载数据集

```
wget -P ~/Downloads https://storage.googleapis.com/cartographer-public-data/bags/backpack_2d/cartographe
```

使用 rosbag 回放一下数据集，检查有哪些传感器：

```
msi@msi:~/carto/carto_ros/data$ rostopic list
/clock
/horizontal_laser_2d
/imu
/rosout
/rosout_agg
/vertical_laser_2d
```

> 可以看到，数据集仅仅包含 imu 传感器，两个激光雷达（水平 + 垂直放置），没有里程计，也没有 TF 变换。
> 疑惑：那么，IMU、激光雷达与机器人本体 base_link 之间的相对位置关系是在哪里确定的呢？[代码中 sensorBridge 需要将传感器的数据转换到机器人本体坐标系下，那么就需要这些相对位置关系]

答：在 carto_ros/src/cartographer_ros/cartographer_ros/urdf/ 文件夹中，有一个 `backpack_2d.urdf`，里面定义了机器人本体与各个传感器之间的相对位置关系，同时，在启动文件中，可以找到如下两行：

```
<param name="robot_description"
  textfile="$(find cartographer_ros)/urdf/backpack_2d.urdf" />

<node name="robot_state_publisher" pkg="robot_state_publisher"
  type="robot_state_publisher" />
```

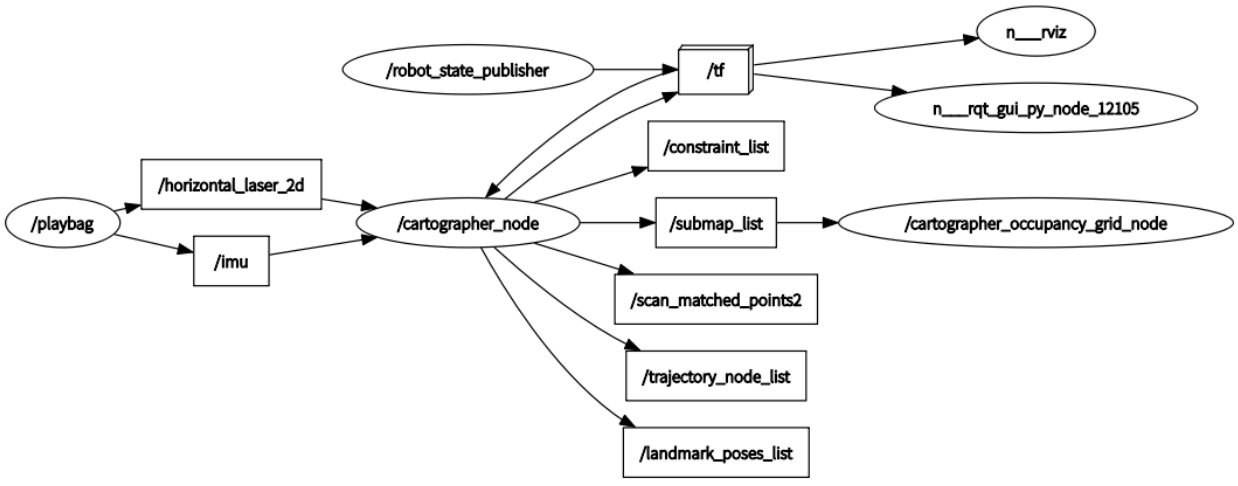
这两行定义了机器人的模型，同时发布了机器人模型中的各个相对坐标关系，在下面的 Demo 运行中，可以看到由 `robot_state_publisher` 节点读取 `backpack_2d.urdf` 模型然后发布的 TF 变换。

1.2. 运行官方给出的 Demo 启动文件

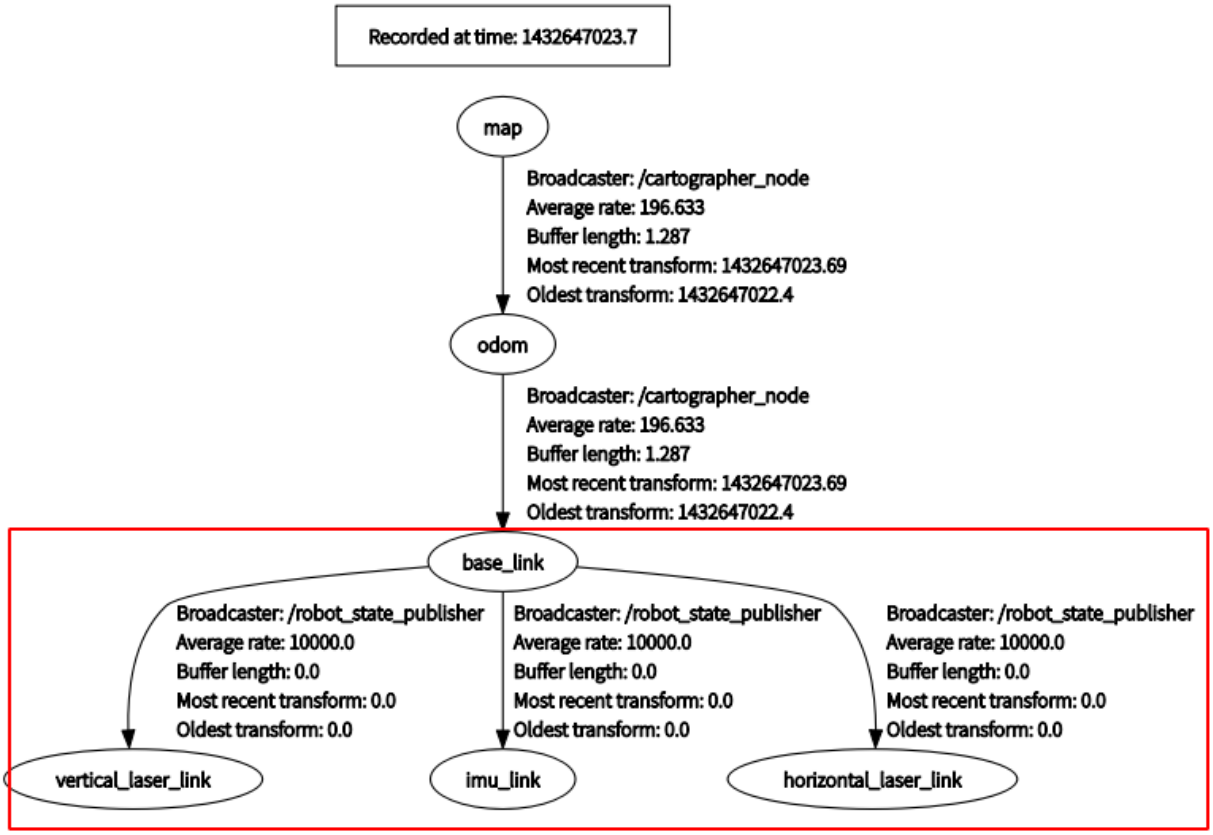
```
source ./install/setup.bash

roslaunch cartographer_ros demo_backpack_2d.launch bag_filename:=${HOME}/Downloads/cartographer_paper_d
```

1.3. rqt 查看节点关系和 TF 变换

1.3.1. 节点关系



1.3.2. TF 变换 (由 `backpack_2d.urdf` 模型定义)

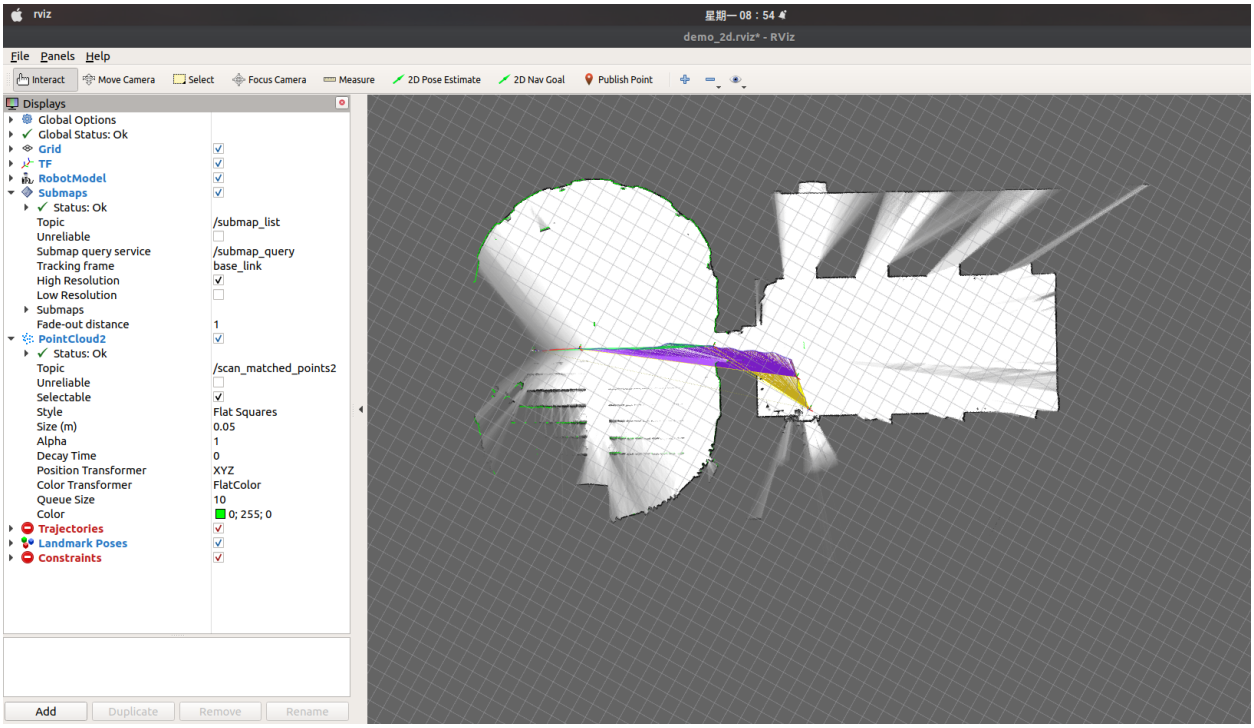


可以看到，红色框标注的地方，就是由 `robot_state_publisher` 节点读取 `backpack_2d.urdf` 模型然后发布的 TF 变换。上面的部分则是由 cartographer 节点定位所发布的 map–>odom 的 TF 变换。

传感器与机器人本体之间的 TF 变换对于运行 cartographer 来说是十分重要的，如果没有这些相对坐标关系，cartographer 将无法运行。

1.4. 运行效果

初始启动:



最终完成:

1.5. 总结

从最终的效果来看，官方 Demo 的效果（建图和跟踪）都比传统的滤波方法好很多，另外，在这个数据集中，仅有激光雷达和 IMU，连里程计都没有使用，这说明了程序对这个数据集做了很多的适配工作。为了验证 cartographer 对其他数据集是否具有普适性，接下来将使用 MIT 数据集进行测试。

2. 运行 MIT 数据集

数据集官方地址：mit stata center data set

Messages logged:

PR2 Message Channel	ROS Message Type	Sensor	Freq
/base_scan	sensor_msgs/LaserScan	Hokuyo UTM-30LX Laser	40Hz
/tilt_scan	sensor_msgs/LaserScan	Hokuyo UTM-30LX Laser	40Hz
/base_odometry/odm	nav_msgs/Odometry	Wheel Odom [Raw]	44Hz
/robot_pose_ekf/odom_combined	geometry_msgs/PoseWithCovarianceStamped	Wheel Odom [Integrated]	25Hz
/torso_lift_imu/data	sensor_msgs/Imu	Microstrain 3DM-GX2 IMU	100Hz
/tf	tf/tfMessage	Various	560Hz

从官方给出的数据集详细来看，有这些传感器 TOpic

- /base_scan: 水平放置的激光雷达
- /tilt_scan: 倾斜放置的激光雷达
- /base_odometry/odom: 里程计
- /robot_pose_ekf/odom_combined: 使用 EFK 滤波得到的里程计数据（可能是里程计 +IMU 融合）
- /torso_lift_imu/data: IMU 数据
- /tf: 传感器相对位置？
- …

2.1. 下载数据集

这个数据集有点坑的地方就是，它的机器人是会坐电梯在楼层之间跑的，因此需要选择一个只在一个平面上跑的数据集。

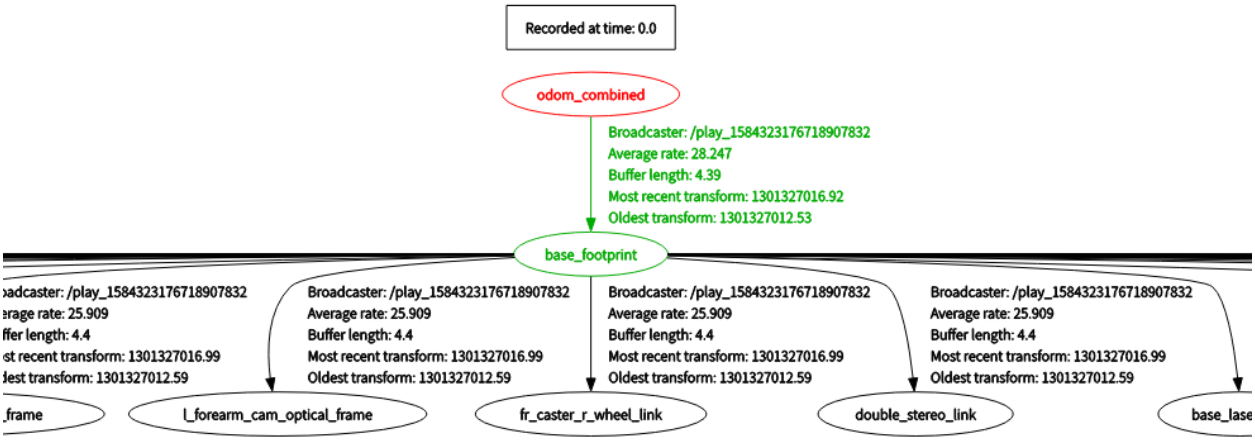
File Listing

Log File	Details	Size (GB)	Time (mins)	Dist (m)	Camera	Floors Visited	Ground Truth
2011-01-18-06-37-58.bag	details	12	5	86	rectified stereo	2 (confirm)	
2011-01-19-07-49-38.bag	details	5	5	68	rectified stereo	2 (confirm)	
2011-01-20-07-18-45.bag	details	5	4	76	rectified stereo	2 (confirm)	
2011-01-20-09-13-31.bag	details	2	2	0	rectified stereo	2 (confirm)	
2011-01-21-09-01-36.bag	details	5	5	87	rectified stereo	2 (confirm)	
2011-01-24-06-18-27.bag	details	1	5	87	throttled stereo	2 (confirm)	
2011-01-25-06-29-26.bag	details	1	5	109	throttled stereo	2 (confirm)	
2011-01-26-06-56-04.bag	details	8	8	194	rectified stereo	2,3 (confirm)	
2011-01-27-07-40-55.bag	details	10	10	222	rectified stereo	2,3 (confirm)	
2011-01-27-07-49-54.bag	details	4	4	94	rectified stereo	2 (confirm)	
2011-01-28-06-37-23.bag	details	9	9	145	rectified stereo	3 (confirm)	
2011-03-11-06-48-23.bag	details	12	12	245	rectified stereo	2 (confirm)	
2011-03-16-06-19-11.bag	details	20	19	320	rectified stereo	9 (confirm)	
2011-03-18-06-00-24.bag	details	17	17	331	rectified stereo	9 (confirm)	
2011-03-18-06-22-35.bag	details	19	7	80	rectified stereo	2 (confirm)	
2011-03-28-08-38-59.bag	details	3	32	712	raw stereo		
2011-04-06-06-38-27.bag	details	5	5	81	raw stereo	2 (confirm)	
2011-04-06-06-56-58.bag	details	3	6	94	throttled stereo	2 (confirm)	
2011-04-06-07-04-17.bag	details	1	6	95	throttled stereo	2 (confirm)	

```
wget http://infinity.csail.mit.edu/data/2011/2011-03-28-08-38-59.bag
```

使用 rosbag 回放，检查数据集内容：

TF 变换信息



> 从 TF 变换树可以看到，这里的里程计信息是采用经过 EKF 滤波之后的里程计坐标系 `odom_combined`，而不是直接的 `odom`。

传感器话题

Topic	Type
/base_odometry/odom	nav_msgs/Odometry
/base_odometry/odometer	pr2_mechanism_controllers/Odometer
/base_odometry/state	pr2_mechanism_controllers/BaseOdometryState
/base_scan	sensor_msgs/LaserScan
/clock	rosgraph_msgs/Clock
/robot_pose_ekf/odom_combined	geometry_msgs/PoseWithCovarianceStamped
/rosout	rosgraph_msgs/Log
/rosout_agg	rosgraph_msgs/Log
/tf	tf/tfMessage
/torso_lift_imu/data	sensor_msgs/Imu
/torso_lift_imu/is_calibrated	std_msgs/Bool
/wide_stereo_throttled/left/camera_info	sensor_msgs/CameraInfo
/wide_stereo_throttled/left/image_raw	sensor_msgs/Image
/wide_stereo_throttled/right/camera_info	sensor_msgs/CameraInfo
/wide_stereo_throttled/right/image_raw	sensor_msgs/Image

2.2. 使用 IMU+ 激光扫描, 运行 cartographer

2.2.1. 配置文件适配

(1).launch 启动文件

在运行之前，需要对配置文件进行修改，因为数据集的传感器 Topic 与官方 Demo 的不一样，主要是 IMU 和激光扫描的 Topic

对 `backpack_2d.launch` 文件进行修改：

修改之后的内容如下：

- 移除了 `robot_state_publisher` 节点以及 `urdf` 机器人模型：这是因为在数据集中已经提供了传感器与机器人本体的相对位置关系
- 传感器 Topic 的重映射：将数据集中的激光扫描 “base_scan” 映射为 “scan”，将 IMU 话题 “torso_lift_imu/data” 映射为 “imu”
- 指定参数配置文件为 `my_script_backpack_2d.lua`

`my_script_backpack_2d.launch`

```
<launch>
  <node name="cartographer_node" pkg="cartographer_ros"
      type="cartographer_node" args="
          -configuration_directory $(find cartographer_ros)/configuration_files
          -configuration_basename my_script_backpack_2d.lua"
      output="screen">
    <remap from="scan" to="base_scan" />
    <remap from="imu" to="torso_lift_imu/data" />

  </node>

  <node name="cartographer_occupancy_grid_node" pkg="cartographer_ros"
      type="cartographer_occupancy_grid_node" args="-resolution 0.05" />
</launch>
```

---

(2).lua 参数配置文件

因为数据集的 TF 变换中，机器人本体坐标系不是 `base_link`，而是 `base_footprint`，因此需要修改。

对 `backpack_2d.lua` 文件进行修改，然后保存为 `my_script_backpack_2d.lua`，主要修改内容：

- 修改 “tracking_frame”，为 “base_footprint”
- 修改 “published_frame”，为 “base_footprint”
- 修改 “num_laser_scans”，为 1 (这是因为数据集中的激光扫描是 LaserScan 消息类型)
- 修改 “num_multi_echo_laser_scans”，“num_point_clouds”，均为 0

`my_script_backpack_2d.lua`:

```
include "map_builder.lua"
include "trajectory_builder.lua"

options = {
  map_builder = MAP_BUILDER,
  trajectory_builder = TRAJECTORY_BUILDER,
  map_frame = "map",
  tracking_frame = "base_footprint",
  published_frame = "base_footprint",        --这个实际上与 `use_odometry`选项无关，
  odom_frame = "odom",           --(如果不使用里程计，直接设置为"odom" 即可)，如果使用里程计，那么将这个设置为
  provide_odom_frame = true,
  publish_frame_projected_to_2d = false,
  use_odometry = false,
  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.use_imu_data = true
TRAJECTORY_BUILDER_2D.num_accumulated_range_data = 1
return options
```

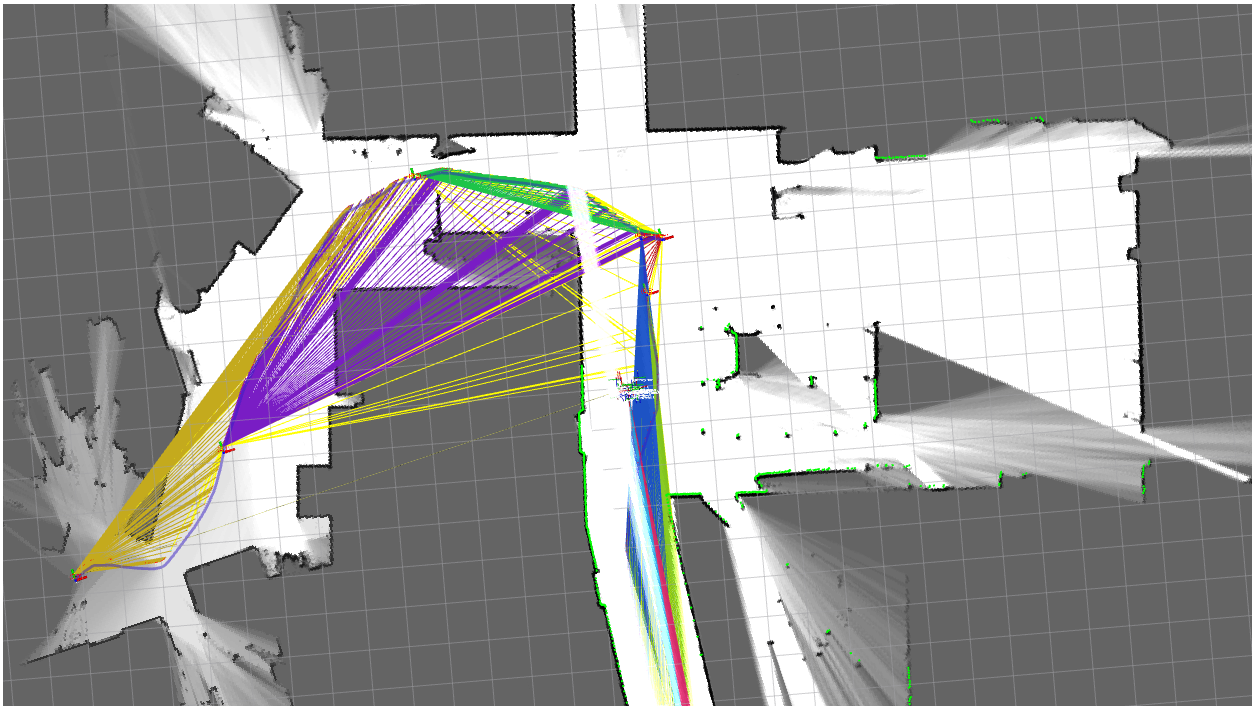
2.2.2. 启动 cartographer

(修改完配置文件之后，需要重新编译一下，否则修改无效)

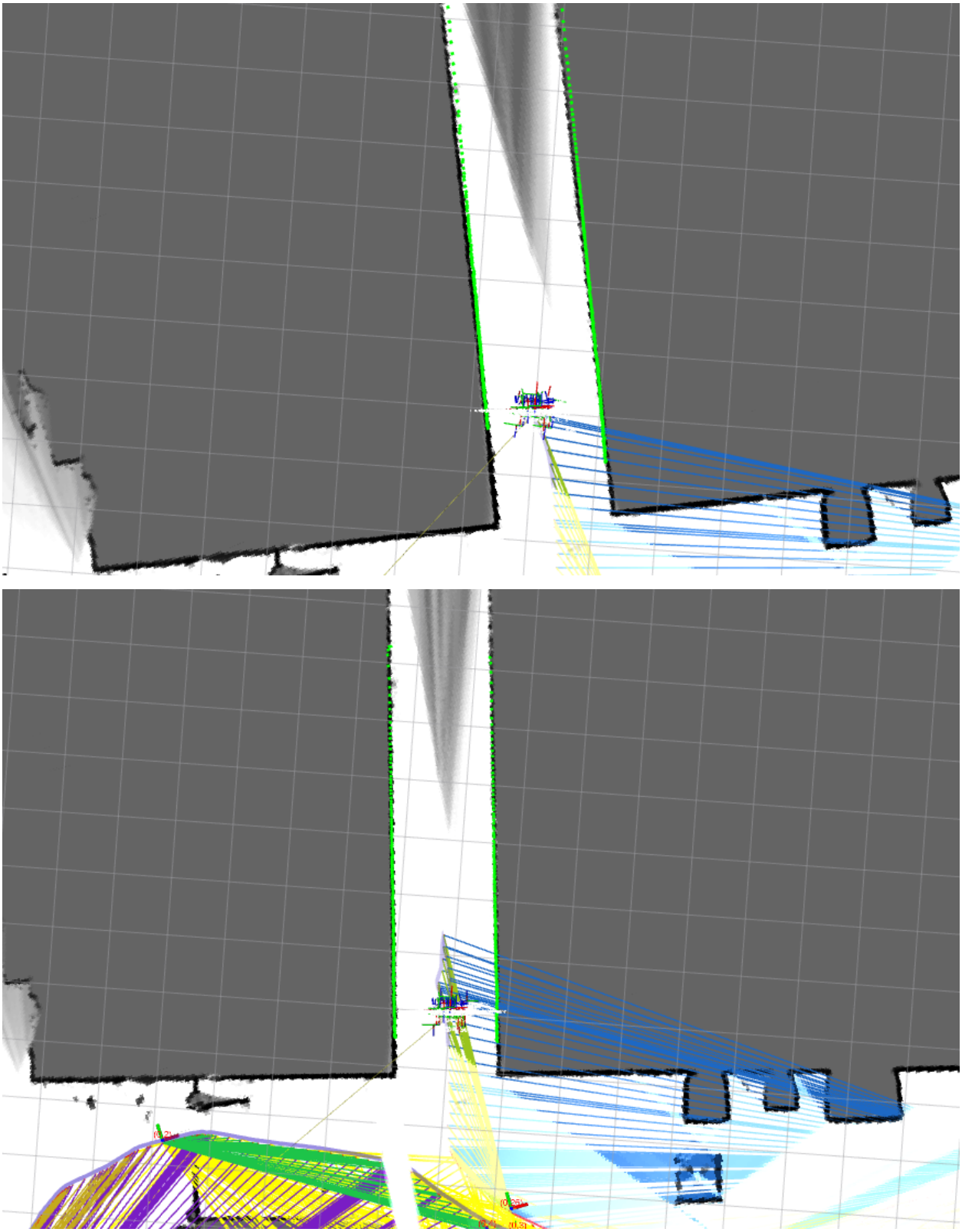
```
source ./install/setup.bash

roslaunch cartographer_ros my_script_launch.launch bag_filename:=/home/msi/carto/carto_ros/data/2011-03-
```

2.2.3. 运行效果

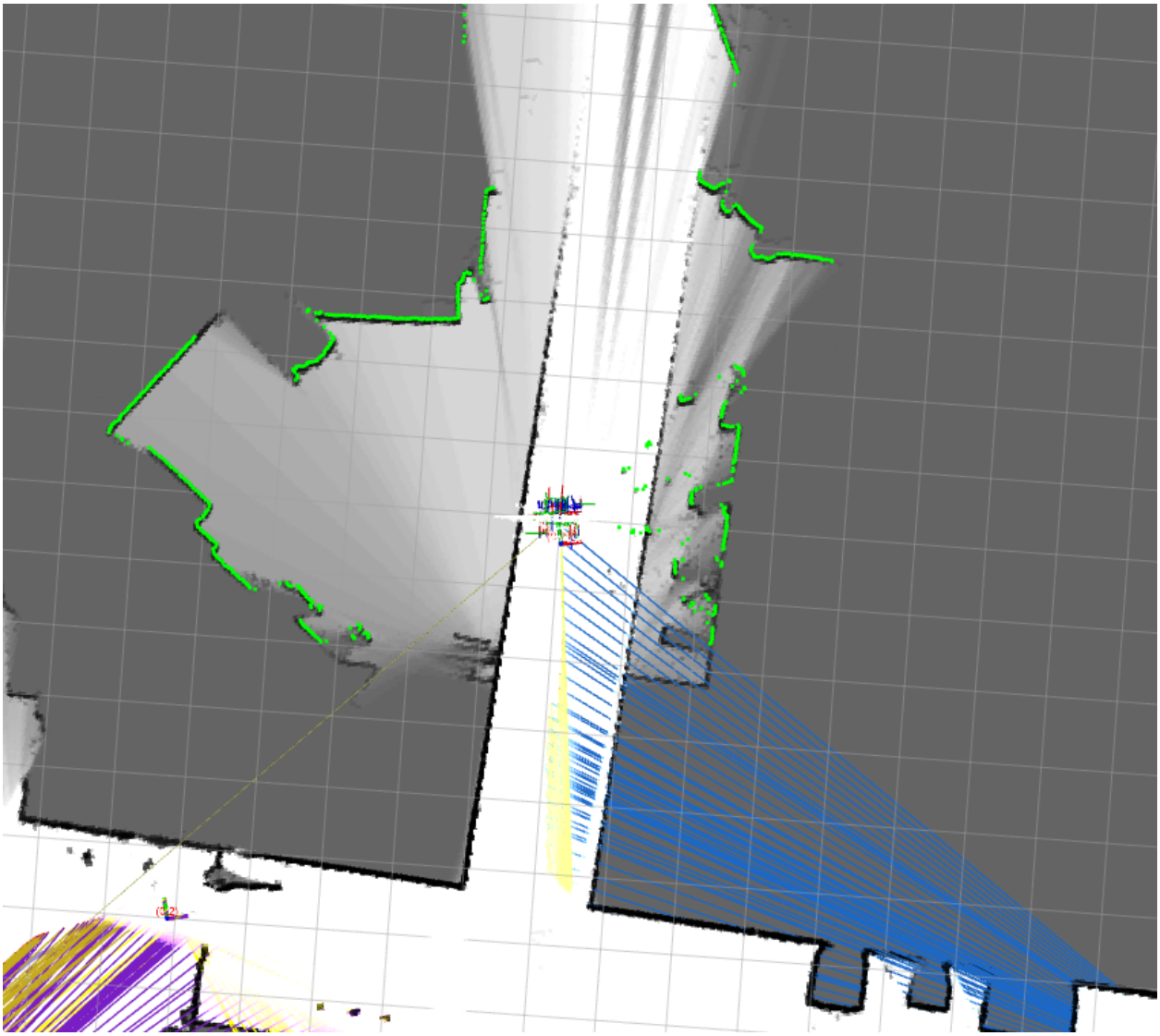
初始启动



遇到长直走廊:



在走廊中定位错误，导致建图重叠：



2.2.4. 总结

单纯使用 IMU+ 激光扫描时，在特征良好的环境下定位、建图效果都不错，但是遇到长直走廊，就出现定位失败的问题，使得建图也出现错误。这种情况下，下面尝试把里程计也融合进去，再看效果如何。

2.3. 使用 IMU+ 里程计 + 激光扫描, 运行 cartographer

2.3.1. 配置文件适配

(1).launch 启动文件，需要对里程计 topic 进行重映射

```
<remap from="odom" to="base_odometry/odom" />
```

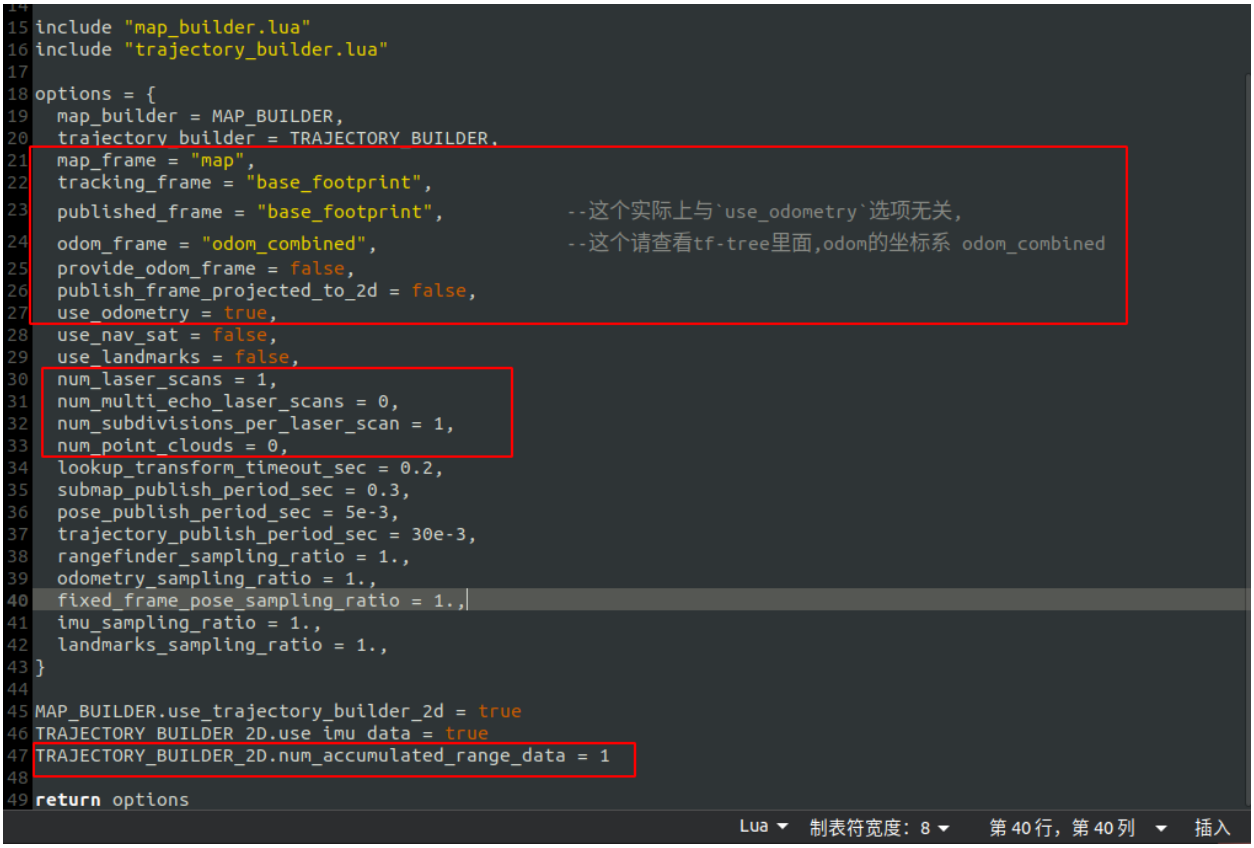
(2).lua 参数配置文件

重点

由于数据集包含的 TF 变换中，里程计的坐标系是 `odom_combined` 而不是 `odom`，（可以往回看看），因此需要修改。

在上面的 IMU 尝试的基础上，继续 `my_script_backpack_2d.lua` 文件进行修改，主要修改内容：

- 修改 “odom_frame”，为 “odom_combined”
- 修改 “provide_odom_frame”，改成 `false`，因为数据集已经提供了里程计的坐标系了
- 修改 “use_odometry”，为 `true`



修改之后的 `my_script_backpack_2d.lua` 完整内容:

```
include "map_builder.lua"
include "trajectory_builder.lua"

options = {
  map_builder = MAP_BUILDER,
  trajectory_builder = TRAJECTORY_BUILDER,
  map_frame = "map",
  tracking_frame = "base_footprint",
  published_frame = "base_footprint",        --这个实际上与 `use_odometry`选项无关,
  odom_frame = "odom_combined",          --这个请查看 tf-tree 里面,odom 的坐标系 odom_combined
  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.use_imu_data = true
TRAJECTORY_BUILDER_2D.num_accumulated_range_data = 1
return options
```

2.3.2. 启动 cartographer

(修改完配置文件之后，需要重新编译一下，否则修改无效)

```
source ./install/setup.bash
```

```
roslaunch cartographer_ros my_script_launch.launch bag_filename:=/home/msi/carto/carto_ros/data/2011-03
```

出现报错

```
at line 126 in /tmp/binarydeb/ros-melodic-tf2-0.6.5/src/buffer_core.cpp
Warning: Invalid argument passed to canTransform argument source_frame in tf2 frame_ids cannot be empty
```

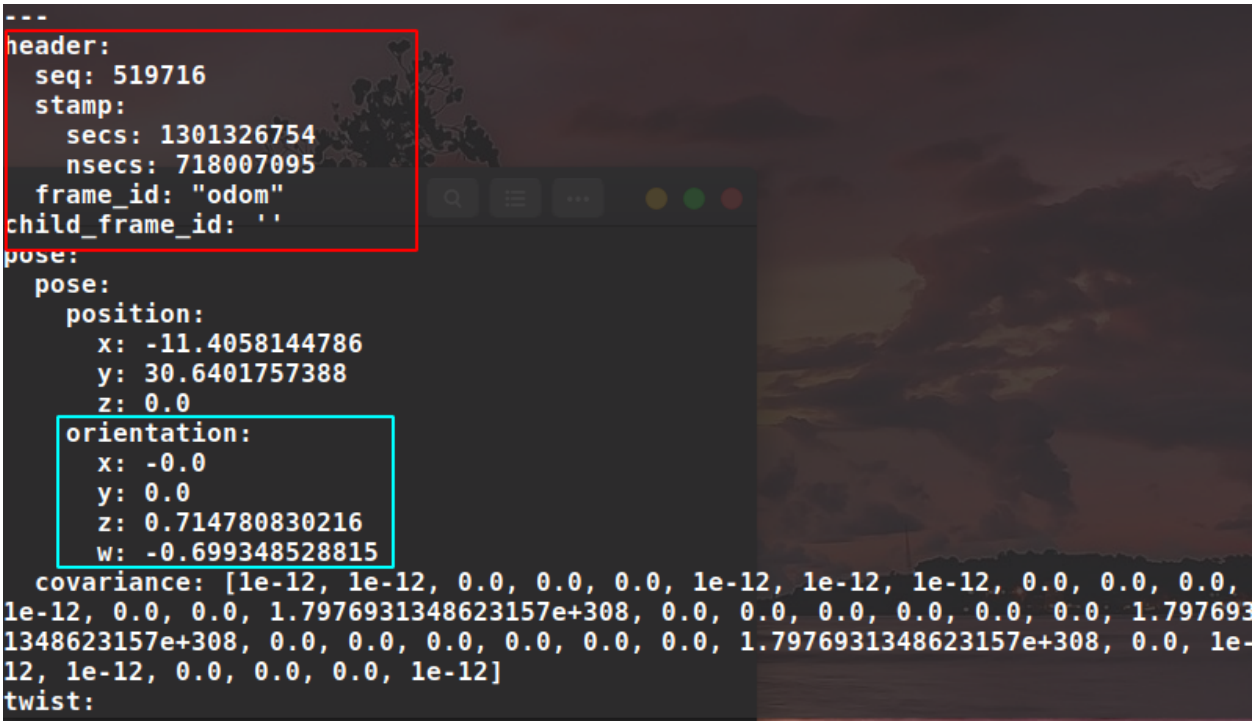
出现这个错误，实际上是因为这个 MIT 数据集的里程计消息不完整

2.3.3. 错误检查

检查里程计消息：

```
rostopic echo /base_odometry/odom
```

得到如下:

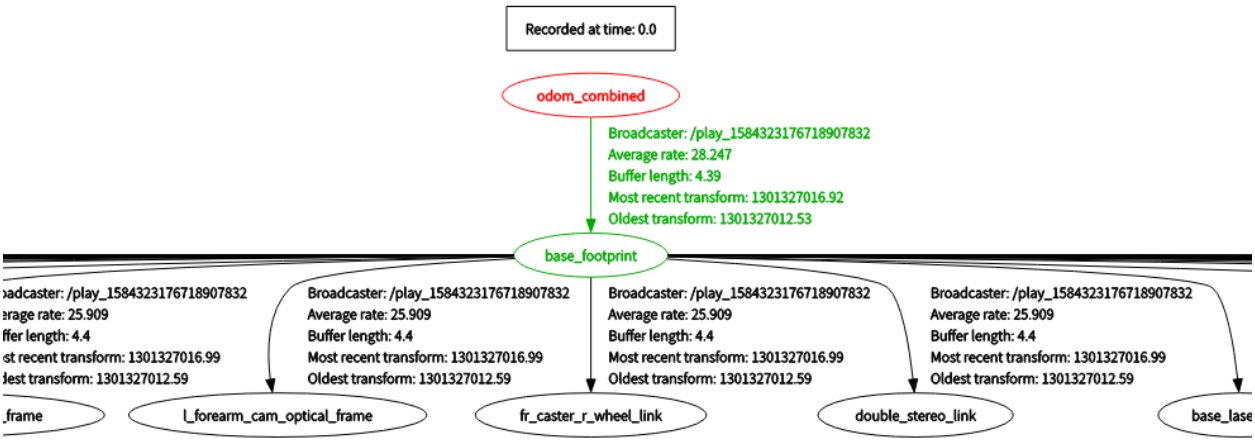


- 问题出现在红色框部分：odom 消息的 `child_frame_id` 居然为空，正常的里程计消息中，`child_frame_id` 一般为 `base_link`，用来指定这是哪个坐标系相对于里程计坐标系的里程信息。
- 蓝色框部分是需要注意的地方，cartographer 要求里程计信息带有四元数旋转信息，这个数据集有，是可以的。

2.3.4. 进一步适配

现在知道了错误所在，那么接下来就是对里程计信息进行修正。

从数据集的 TF 变换：



可以看到，这里的里程计坐标系是采用经过 EKF 滤波之后的里程计坐标系 `odom_combined`，而不是直接的 `odom`。

那么干脆使用经过 EKF 滤波之后融合里程计/robot_pose_ekf/odom_combined 好了，但是问题来了，这个 msg 的消息类型是 geometry_msgs/PoseWithCovarianceStamped 而不是 nav_msgs/Odometry

接下来需要编写一个节点，订阅/robot_pose_ekf/odom_combined 这个 Topic，然后发布 nav_msgs/Odometry 类型的里程 msg

这个节点我已经写好了

Pose2Odom.cpp

```
#include <ros/ros.h>
#include <message_filters/subscriber.h>
#include <message_filters/synchronizer.h>
#include <message_filters/sync_policies/approximate_time.h>
#include <nav_msgs/Odometry.h>
#include <nav_msgs/Path.h>
#include <sensor_msgs/Imu.h>
#include <sensor_msgs/NavSatFix.h>
#include <geometry_msgs/Vector3.h>
#include <geometry_msgs/PoseWithCovarianceStamped.h>

#include <tf/tf.h>
#include <tf/transform_broadcaster.h>
#include <boost/array.hpp>
ros::NodeHandle *nh;
tf::TransformBroadcaster *tfb_;
ros::Publisher *odom_pub;                              //odom 发布器

void odom_callback(const geometry_msgs::PoseWithCovarianceStampedPtr& odomMsg)
{

  nav_msgs::Odometry msg;
  msg.header=odomMsg->header;
  msg.pose.pose=odomMsg->pose.pose;
  msg.child_frame_id="base_footprint";
  odom_pub->publish(msg);
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "Pose2Odom");
    nh=new ros::NodeHandle("");
    tfb_=new tf::TransformBroadcaster();
    odom_pub=new ros::Publisher;
    *odom_pub=nh->advertise<nav_msgs::Odometry>("odom",10,true);

    //订阅里程计
    message_filters::Subscriber<geometry_msgs::PoseWithCovarianceStamped> odom_sub(*nh, "/robot_pose_ek

    odom_sub.registerCallback(&odom_callback);

    //消息同步
//    typedef message_filters::sync_policies::ApproximateTime<nav_msgs::Odometry, nav_msgs::Odometry> M
//    message_filters::Synchronizer<MySyncPolicy> sync(MySyncPolicy(10), odom_sub, ekf_sub);//queue siz
//    sync.registerCallback(&callback/*boost::bind(&callback, _1, _2)*/);
```

```
    ROS_INFO("Msg block control node start");
    ros::spin();
}
```

这个节点的主要功能是：

- 订阅 “/robot_pose_ekf/odom_combined”，然后转换成 “nav_msgs::Odometry” 消息类型
- 同时，设置 `child_frame_id` 为:`base_footprint`

2.3.5. 再次启动 cartographer

(1) 把`.launch` 里面的关于里程计的 reamap 去掉，因为上面的节点直接发布 Topic 为 `odom` 的消息，就不需要重映射了。

(2) 启动上面的 `Pose2Odom` 节点

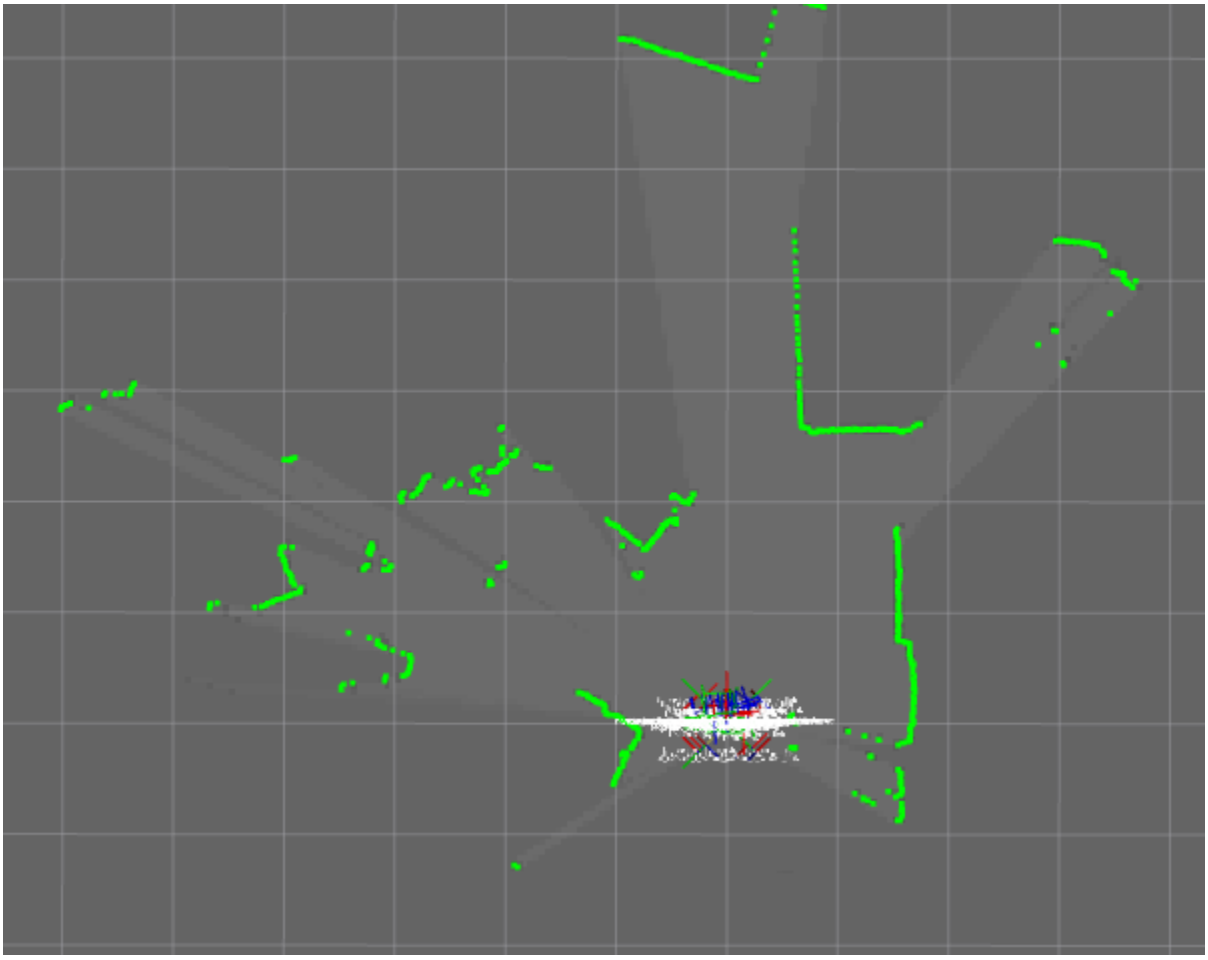
```
msi@msi:~/ros_workspace$ rosrun Pose2Odom Pose2Odom
[ INFO] [1584328528.064815042]: Pose2Odom node start
```

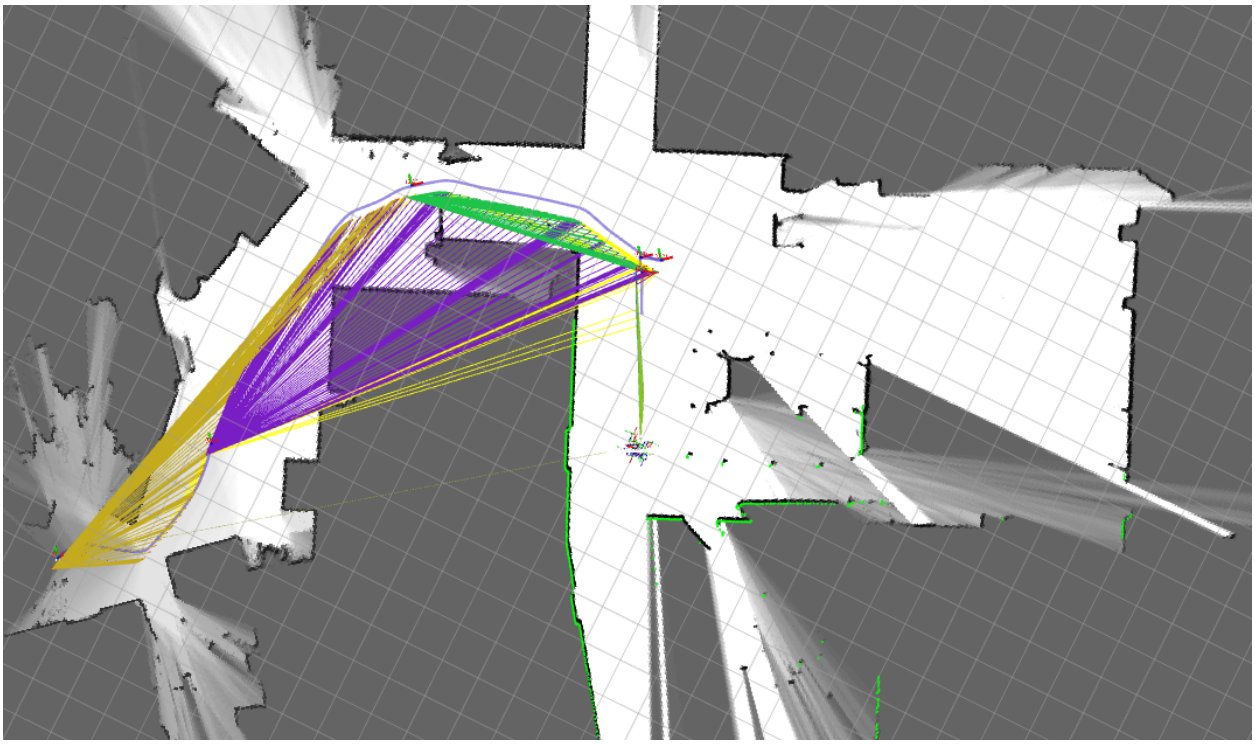
(3) 启动 cartographer

```
source ./install/setup.bash

roslaunch cartographer_ros my_script_launch.launch bag_filename:=/home/msi/carto/carto_ros/data/2011-03-
```

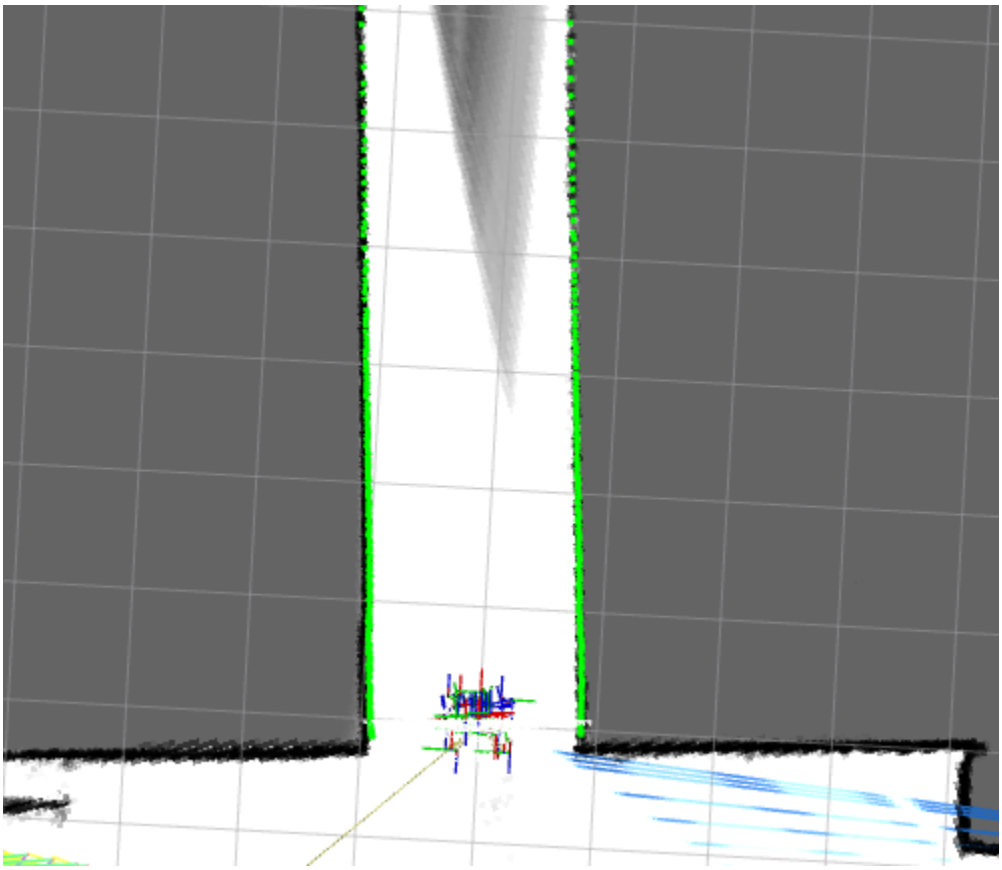
2.3.6. 运行效果

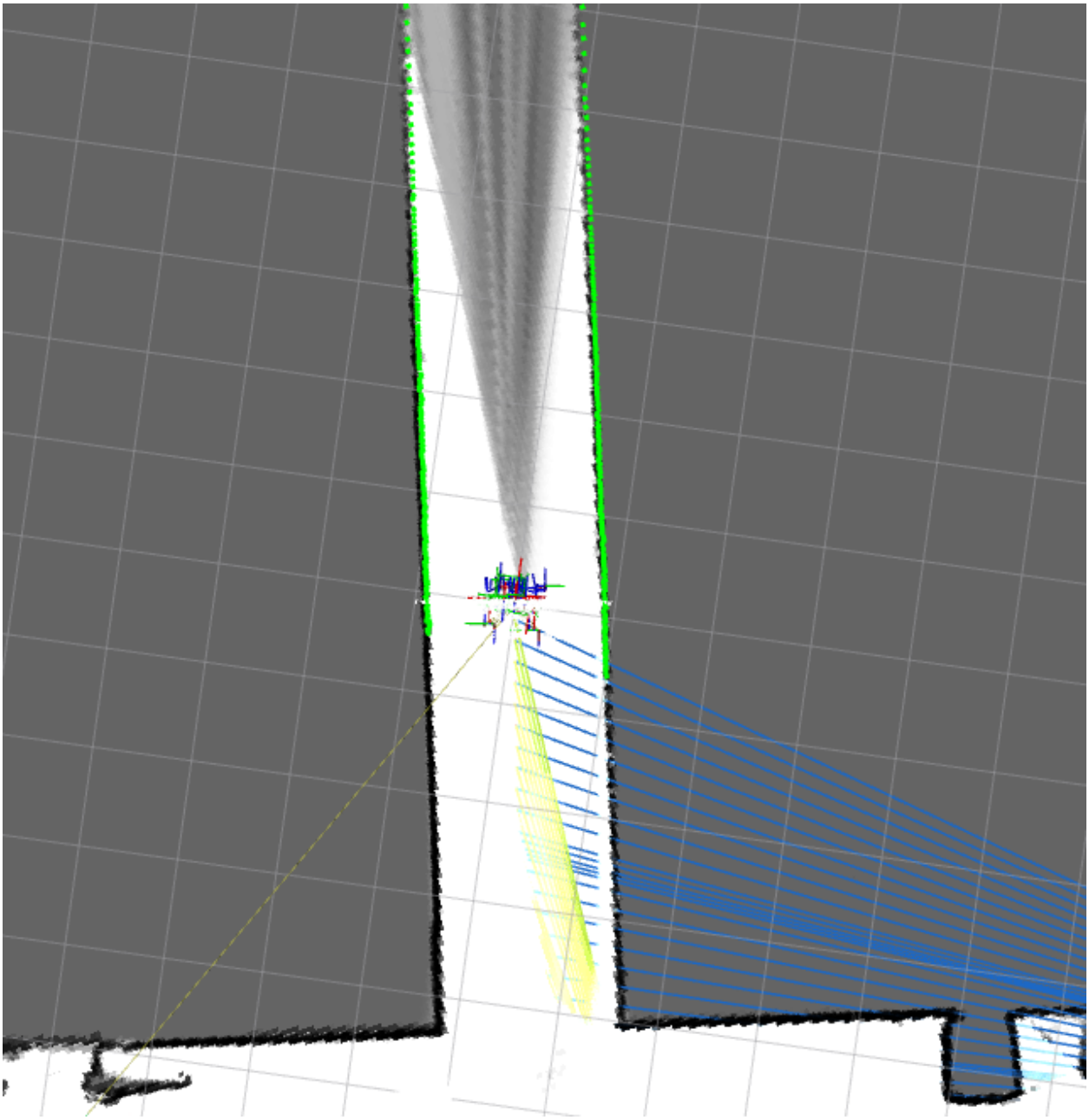
初始运行 (可以正常启动了)



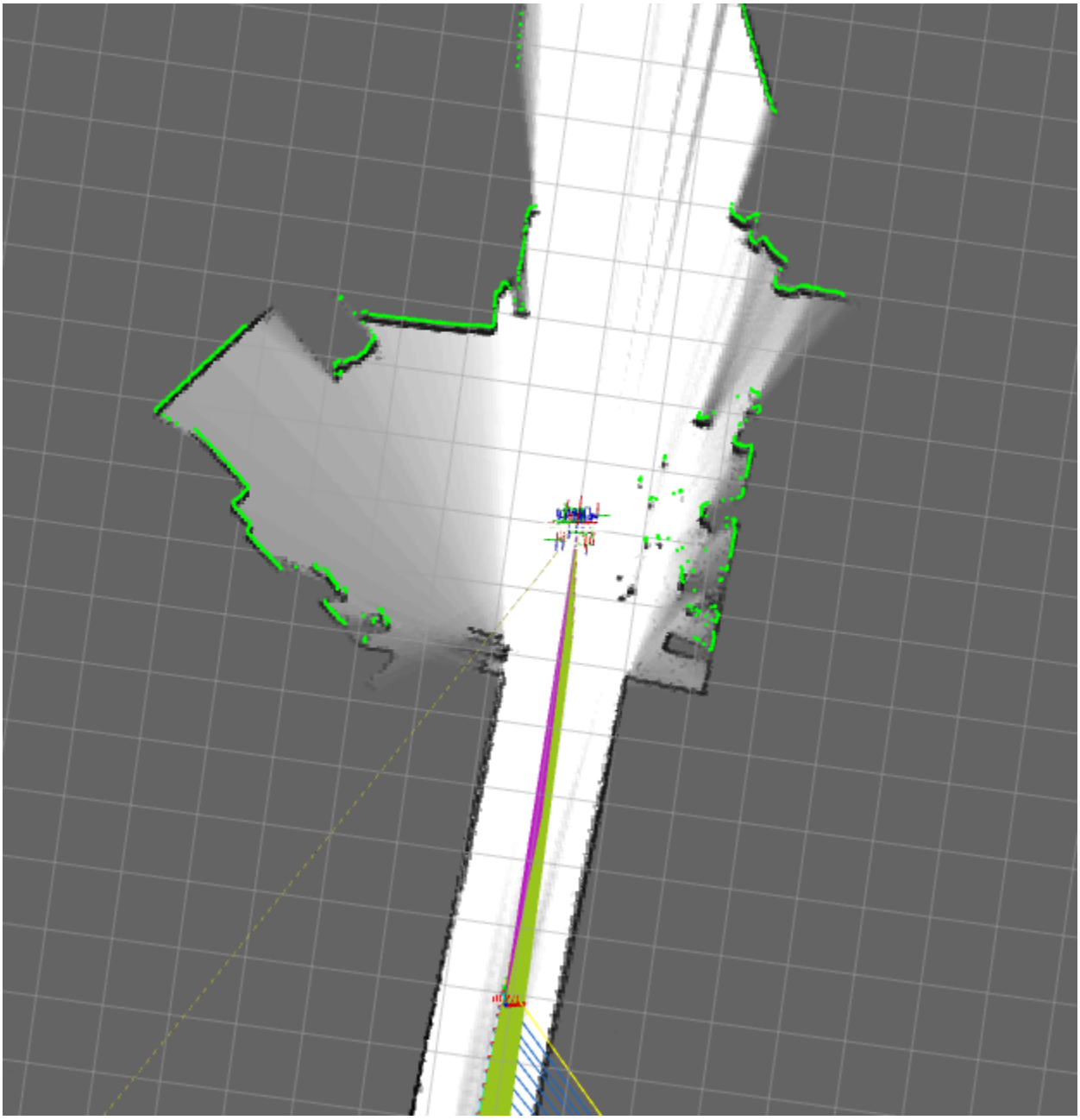


再次遇到长直走廊

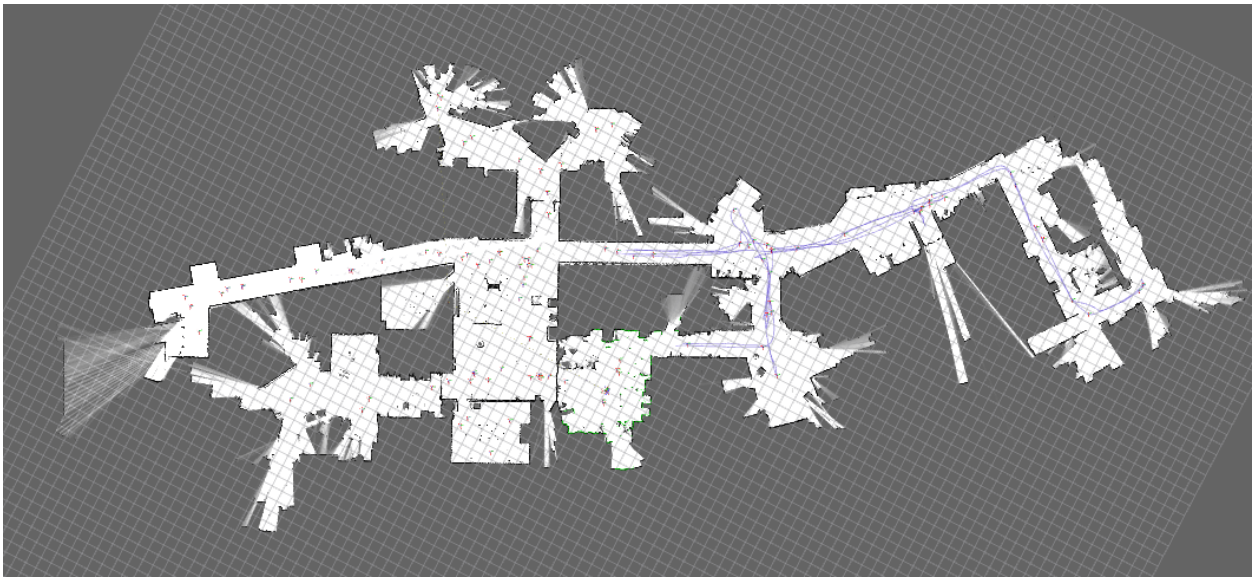




没有出现地图重叠的情况了



最终建立完整的地图:



2.4. 总结

通过使用 MIT 数据集进行测试，可以发现，cartographer 具有一定的普适性，但是对于长直走廊时的建图和定位，cartographer 其实是会受到影响的，单纯使用 IMU+ 激光扫描会出现定位失败，从而导致建图错误的情况，融合里程计之后，情况有所改善 (这是因为 cartographer 里面关于位姿图的优化问题中添加了里程计残差项作为约束)，同时里程计也为姿态外推器 (航迹推算) 提供了比较好的先验信息 (因为在较短时间内，里程计数据可看作是准确的)。

3. 一些思考

3.1. cartographer 已经实现的

(1) 定位方式有 3 种

- correlative_scan_matching: ～公司采用的
- fast_correlative_scan_matching: 分枝定界加速相关性匹配
- ceres_scan_matching: 非线性优化位姿

(2) 图优化理论

Pose_Graph: 位姿图

顶点：

- 子图全局位姿
- 节点全局位姿

约束项 (残差项):

- 节点与当前子图的 `correlative_scan_matching` 可产生一个节点--子图约束
- 节点与所有子图的 `fast_correlative_scan_matching` 可产生多个节点--子图约束 (即回环检测，如果回环匹配上了，则可构成约束)
- 节点之间的里程计信息产生节点--节点约束
- 节点之间的局部 SLAM(local SLAM: 即节点与子图 `correlative_scan_matching` 得到节点在 local map 的位姿) 可构成节点--节点约束 [利用两个节点的 local map 位姿产生]

- 路标点，产生路标--节点的约束？(这一块代码还没详细阅读)

3.2. cartographer 还没有实现的

- GPS 数据的融合 (用于重定位、检测回环、产生约束)
- 重定位功能
- 视觉信息的融合

3.3. 可以有突破点的地方

- 基于路标的长直走廊定位 (基于视觉的路标信息)
- 重定位
- GPS 融合