

南海鲨哨兵导航调试v1.0.5

1二维导航

1.1 前言

首先明确概念，“二维导航”是指在二维传感器下进行导航任务执行所使用的导航框架，与导航具体的框架无关，也就是说我们既可以用二维传感器，也可以使用三维传感器来进行导航，这里只是为了称呼方便称为二维导航，同时我也建议大多数导航的新手在学习使用导航框架时先从二维导航入手，因为现有的导航框架无论是Nav1还是Nav2都是以二维地图作为基础进行导航任务的，而大多数Rm赛场上也是使用二维地图作为执行导航任务，所以二维传感器在精度允许的条件下可以起到删繁就简的效果，为后续开发以三维传感器（比如大多数战队使用的Mid360）支持导航起到很好的铺垫作用

而在这里由于远古代码的丢失，我只能凭借记忆和老文档进行一部分内容的提示，如果有需要的话就需要后续的队员完善了

Note

需要注意的是，这份文件更多是调试时候需要注意的重点的提醒和解释说明，并不会从0开始说明如何书写一份能完成导航任务的代码，所以阅读时一定要结合附带的代码以及论文一并食用

1.2 二维导航系统的简单介绍

1.2.1 2d slam_基于gmapping算法

利用github开源的[scan_tool:CCNYRoboticsLab/scan_tools: ROS Laser scan tools \(github.com\)](#)以及ros自带的gmapping进行2d雷达建图

参考博客：

[ROS下：无里程计 仅有rplidar A2激光雷达+laser_scan_tools运行gmapping-CSDN博客](#)
[Rplidar 报错提示：Error,_operation time out. RESULT_OPERATION_TIMEOUT!-CSDN博客](#)

下载必要的依赖 使用命令：

```
sudo apt-get install ros-noetic-csm  
sudo apt-get install ros-noetic-gmapping
```

重点在于修改.launch文件为如下形式

```
<!--
Example launch file: uses laser_scan_matcher together with
slam_gmapping
-->

<launch>

#### set up data playback from bag #####
<param name="/use_sim_time" value="false"/><!-- 因为Gmapping 的
simulation 时间是True, 改为false 网上查到的-->

#### rplidar_a1 #####
<!--激光雷达的启动文件-->
<node name="rplidarNode"           pkg="rplidar_ros"
      type="rplidarNode" output="screen">
    <param name="serial_port"        type="string"
          value="/dev/ttyUSB0"/>
    <param name="serial_baudrate"    type="int"     value="115200"/><!--
A1 -->

    <param name="frame_id"          type="string" value="laser"/>
    <param name="inverted"          type="bool"   value="false"/>
    <param name="angle_compensate"   type="bool"   value="true"/>
    <param name="scan_mode"         type="string" value="Sensitivity"/>
</node>

#### publish an example base_link -> laser transform #####
<node pkg="tf" type="static_transform_publisher"
      name="base_link_to_laser"
      args="0.0 0.0 0.0 0.0 0.0 0.0 /base_link /laser 40" />

#### start rviz #####
<node pkg="rviz" type="rviz" name="rviz"
      args="-d $(find laser_scan_matcher)/demo/demo_gmapping.rviz"/>

#### start the laser scan_matcher #####
<node pkg="laser_scan_matcher" type="laser_scan_matcher_node"
      name="laser_scan_matcher_node" output="screen">

  <param name="fixed_frame" value = "/odom"/>
```

```

<param name="max_iterations" value="10"/>

<param name="base_frame" value = "/base_link"/>
<param name="use_odom" value="false"/>
<param name="publy_pose" value = "true"/>
<param name="publy_tf" value="true"/>

</node>

##### start gmapping #####
<!--前三个param必须设置修改，要不然tf_tree不完整-->
<node pkg="gmapping" type="slam_gmapping" name="slam_gmapping"
output="screen">
    <param name="base_frame" value="/base_link"/> <!--***机器人的坐标系-->
    <param name="odom_frame" value="/odom" /> <!--***世界坐标系-->
    <param name="map_frame" value="/map" /> <!--***地图坐标系-->

    <param name="map_update_interval" value="1.0"/>
    <param name="maxUrange" value="5.0"/>
    <param name="sigma" value="0.1"/>
    <param name="kernelSize" value="1"/>
    <param name="lstep" value="0.15"/>
    <param name="astep" value="0.15"/>
    <param name="iterations" value="1"/>
    <param name="lsigma" value="0.1"/>
    <param name="ogain" value="3.0"/>
    <param name="lskip" value="1"/>
    <param name="srr" value="0.1"/>
    <param name="srt" value="0.2"/>
    <param name="str" value="0.1"/>
    <param name="stt" value="0.2"/>
    <param name="linearUpdate" value="1.0"/>
    <param name="angularUpdate" value="0.5"/>
    <param name="temporalUpdate" value="0.4"/>
    <param name="resampleThreshold" value="0.5"/>
    <param name="particles" value="10"/>
    <param name="xmin" value="-5.0"/>
    <param name="ymin" value="-5.0"/>
    <param name="xmax" value="5.0"/>
    <param name="ymax" value="5.0"/>
    <param name="delta" value="0.02"/>
    <param name="llsamplerange" value="0.01"/>
    <param name="llsamplestep" value="0.05"/>
    <param name="lasamplerange" value="0.05"/>
    <param name="lasamplestep" value="0.05"/>

```

```
</node>
```

```
</launch>
```

运行时要使用命令更改权限

```
sudo chmod 777 /dev/ttyUSB0
```

运行命令行：

```
roslaunch laser_scan_matcher demo_gmapping.launch
```

建图效果：

在线：<https://wv9y1mjo8hb.feishu.cn/file/Z4zzbUFu3oIDF9xGdMtcJnC6np>

1.2.2 参考网站：

1, ros navigation部分理解

1>[ROS 2D导航原理系列教程合集 \(WHEELTEC\) - 哔哩哔哩_bilibili](#)

2>[ROS与navigation教程-编写自定义全局路径规划 - 创客智造/爱折腾智能机器人\(ncynl.com\)](#)

3>[慢慢的回味 - Continues Learning_\(liuxiaofei.com.cn\)](#)

4>[navigation - ROS Wiki](#)

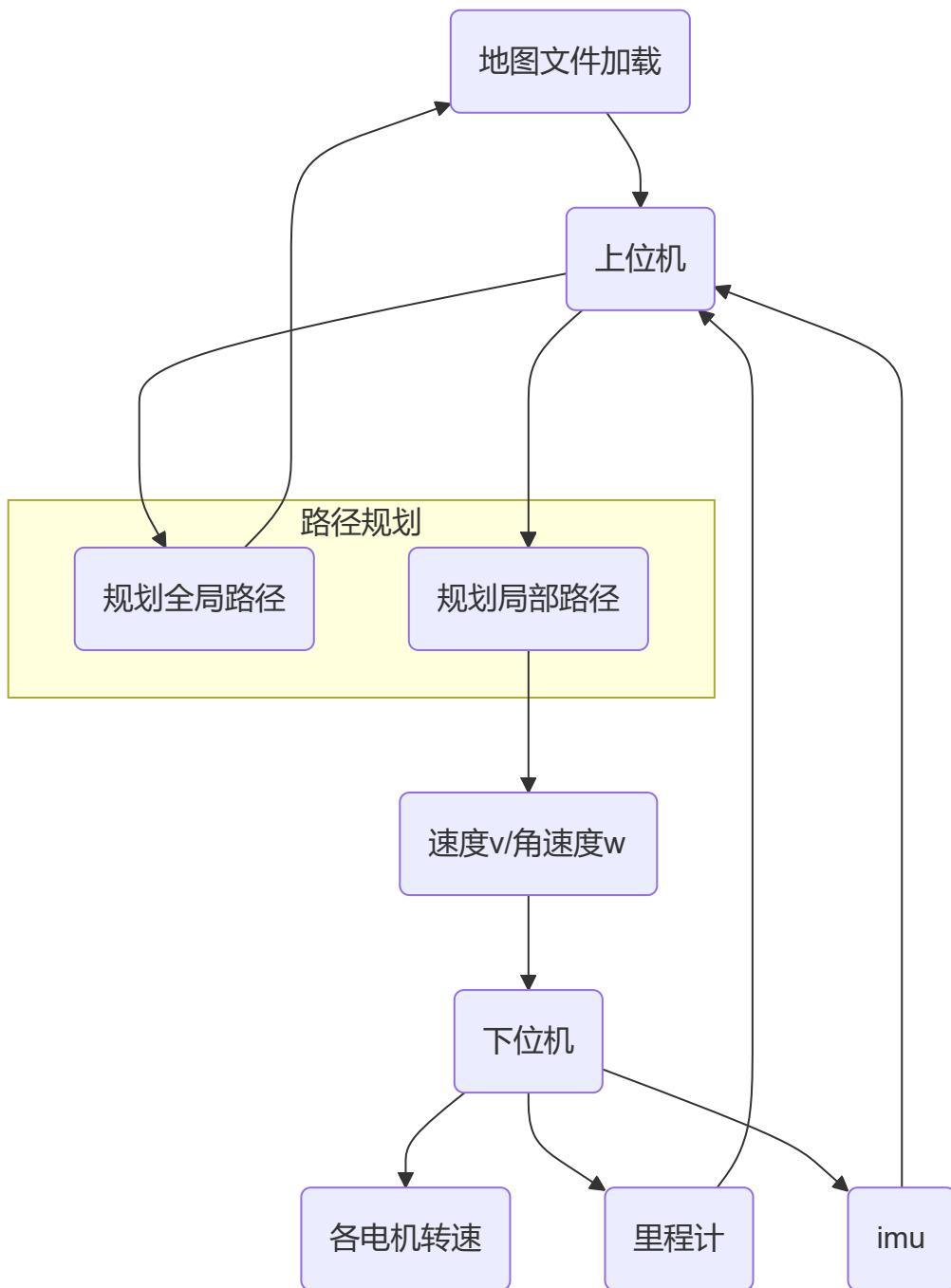
5>[ROS：导航功能详解_ros导航_Hello xiǎo lěi的博客-CSDN博客](#)

2, 上下位机通信原理理解

1>[上位机与下位机通信 - 慢慢的回味_\(liuxiaofei.com.cn\)](#)

2>[pwm信号控制舵机的简单原理_pwm控制舵机的工作原理-CSDN博客](#)

3》[Development-Board-C-Examples/RoboMaster开发板C型嵌入式软件教程文档.pdf at master · RoboMaster/Development-Board-C-Examples \(github.com\)](#)

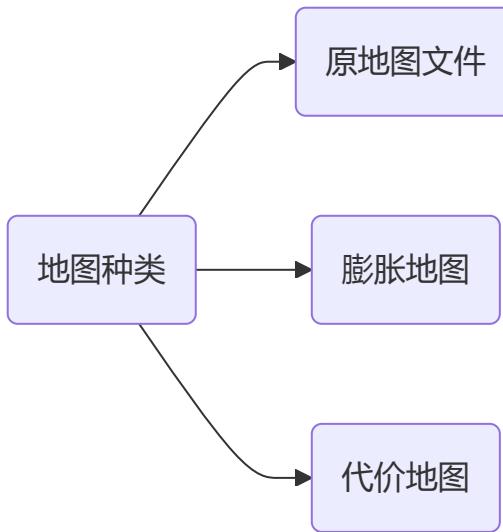


1.2.3 地图的加载

```
sudo apt install ros-<ROS版本>-map-server
```

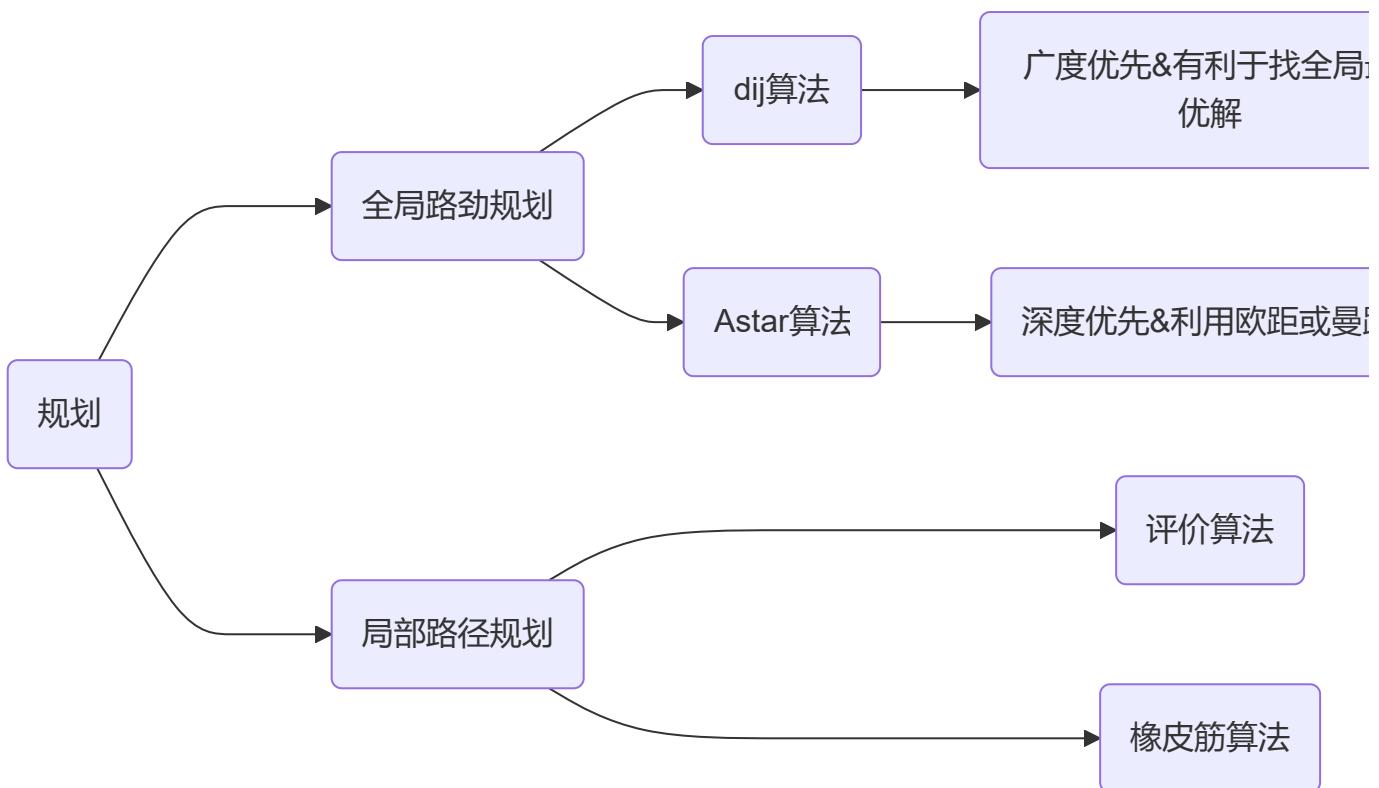
地图保存需要编写launch文件

```
<launch>
  <arg name="filename" value="$(find nva_demo)/map/nav" />
  <node name="map_save" pkg="map_server" type="map_saver" args="-f
$arg filename" />
</launch>
```



1.2.4 acml的使用

```
sudo apt install ros-<ROS版本>-navigation
```



路径:

- /opt/ros/noetic/include/global_planner/dijkdtra.h
- /opt/ros/noetic/include/global_planner/astar.h

1.2.5 tf_node结点的编写

由于部分情况下机器人的雷达位置与底盘位置不重合，所以很多时候要考虑使用tf转换，由于实验用车雷达和底盘位置差距不大，应该不会产生路径规划上的问题，这里暂时不考虑

/参考gmapping文件的tf信息

```
#### publish an example base_link -> laser transform #####
<node pkg="tf" type="static_transform_publisher" name="base_link_to_laser"
args="0.0 0.0 0.0 0.0 0.0 0.0 /base_link /laser 40" />
```

1.2.6 编写local_planner插件

需要编写一段local_planner类的代码，规划局部路径

1.3 navigation的调试

[navigation/Tutorials/Navigation Tuning Guide - ROS Wiki](#)

关键参考博客 [【精选】ROS 导航安装及实现（二十二）ros map_server源码安装啥也不是的py人的博客-CSDN博客](#)

这里关键在于对两个package的编写，amcl和move_base包，

Ø amcl包的作用在于根据map_server的信息和雷达的实时信息判断出当前小车所在的位置

amcl包有许多参数：

```
<launch>
<node pkg="amcl" type="amcl" name="amcl" output="screen">
  <!-- Publish scans from best pose at a max of 10 Hz -->
  <param name="odom_model_type" value="omni"/>
  <param name="odom_alpha5" value="0.1"/>
  <param name="gui_publish_rate" value="10.0"/>
  <param name="laser_max_beams" value="30"/>
  <param name="min_particles" value="500"/>
  <param name="max_particles" value="5000"/>
  <param name="kld_err" value="0.05"/>
  <param name="kld_z" value="0.99"/>
  <param name="odom_alpha1" value="0.2"/>
  <param name="odom_alpha2" value="0.2"/>  <!-- translation std dev, m
-->
  <param name="odom_alpha3" value="0.8"/>
  <param name="odom_alpha4" value="0.2"/>
  <param name="laser_z_hit" value="0.5"/>
  <param name="laser_z_short" value="0.05"/>
```

```

<param name="laser_z_max" value="0.05"/>
<param name="laser_z_rand" value="0.5"/>
<param name="laser_sigma_hit" value="0.2"/>
<param name="laser_lambda_short" value="0.1"/>
<param name="laser_model_type" value="likelihood_field"/> <!--
<param name="laser_model_type" value="beam"/> -->
<param name="laser_likelihood_max_dist" value="2.0"/>
<param name="update_min_d" value="0.2"/>
<param name="update_min_a" value="0.5"/>
<param name="odom_frame_id" value="odom"/>
<param name="resample_interval" value="1"/>
<param name="transform_tolerance" value="0.1"/>
<param name="recovery_alpha_slow" value="0.0"/>
<param name="recovery_alpha_fast" value="0.0"/>
<param name="use_map_topic" value="true"/>
<param name="base_frame_id" value="/base_link" />
<param name="odom_frame_id" value="/odom" />
</node>
</launch>

```

以下几点需要注意：

```
<param name="odom_model_type" value="omni"/>
```

这个参数有diff和omni两种选项

```
<param name="base_frame_id" value="/base_link" />
<param name="odom_frame_id" value="/odom" />
```

这里是决定后面tf_tree长相的话题名

 move_base的作用在于根据定位,里程计、tf、目标点的信息决定给小车的速度信息

move_base包能否正常运行取决于两个决定性因素：

- 1，四个地图、规划文件的正确编写，包含在path.launch文件中：

```

<launch>
  <node pkg = "move_base" type="move_base" respawn="false"
name="move_base" output="screen" clear_params="true">
    <rosparam file="/home/nvidia/RPLidar_gmapping_ws/src/scan_tools-
indigo/laser_scan_matcher/demo/costmap_common_params.yaml"
command="load" ns="global_costmap"/>

```

```

<rosparam file="/home/nvidia/RPLidar_gmapping_ws/src/scan_tools-
indigo/laser_scan_matcher/demo/costmap_common_params.yaml"
command="load" ns="local_costmap"/>
<rosparam file="/home/nvidia/RPLidar_gmapping_ws/src/scan_tools-
indigo/laser_scan_matcher/demo/local_costmap_params.yaml"
command="load"/>
<rosparam file="/home/nvidia/RPLidar_gmapping_ws/src/scan_tools-
indigo/laser_scan_matcher/demo/global_costmap_params.yaml"
command="load"/>
<rosparam file="/home/nvidia/RPLidar_gmapping_ws/src/scan_tools-
indigo/laser_scan_matcher/demo/base_local_planner_params.yaml"
command="load"/>
</node>
</launch>

```

- 2, tf树的形状正常：由根到叶应该是： /map->/odom->/base_footprint->/base_link->/laser其中/map->/odom是amcl包中自动生成实时变化的
最后整体的launch文件长这样：

```

<launch>
  <arg name = "map" default="nav.yaml"/>
  <node name = "map_server" pkg="map_server" type="map_server"
args="/home/nvidia/Desktop/nav/$(arg map)"/>    <node pkg="tf"
type="static_transform_publisher" name="odom_to_base_footprint"
  args="0.0 0.0 0.0 0.0 0.0 0.0 /odom /base_footprint 40" />
  <node pkg="tf" type="static_transform_publisher"
name="base_footprint_to_base_link"
  args="0.0 0.0 0.0 0.0 0.0 0.0 /base_footprint /base_link 40" />
  <node pkg="tf" type="static_transform_publisher"
name="base_link_to_laser"
  args="0.0 0.0 0.0 0.0 0.0 0.0 /base_link /laser 40" />

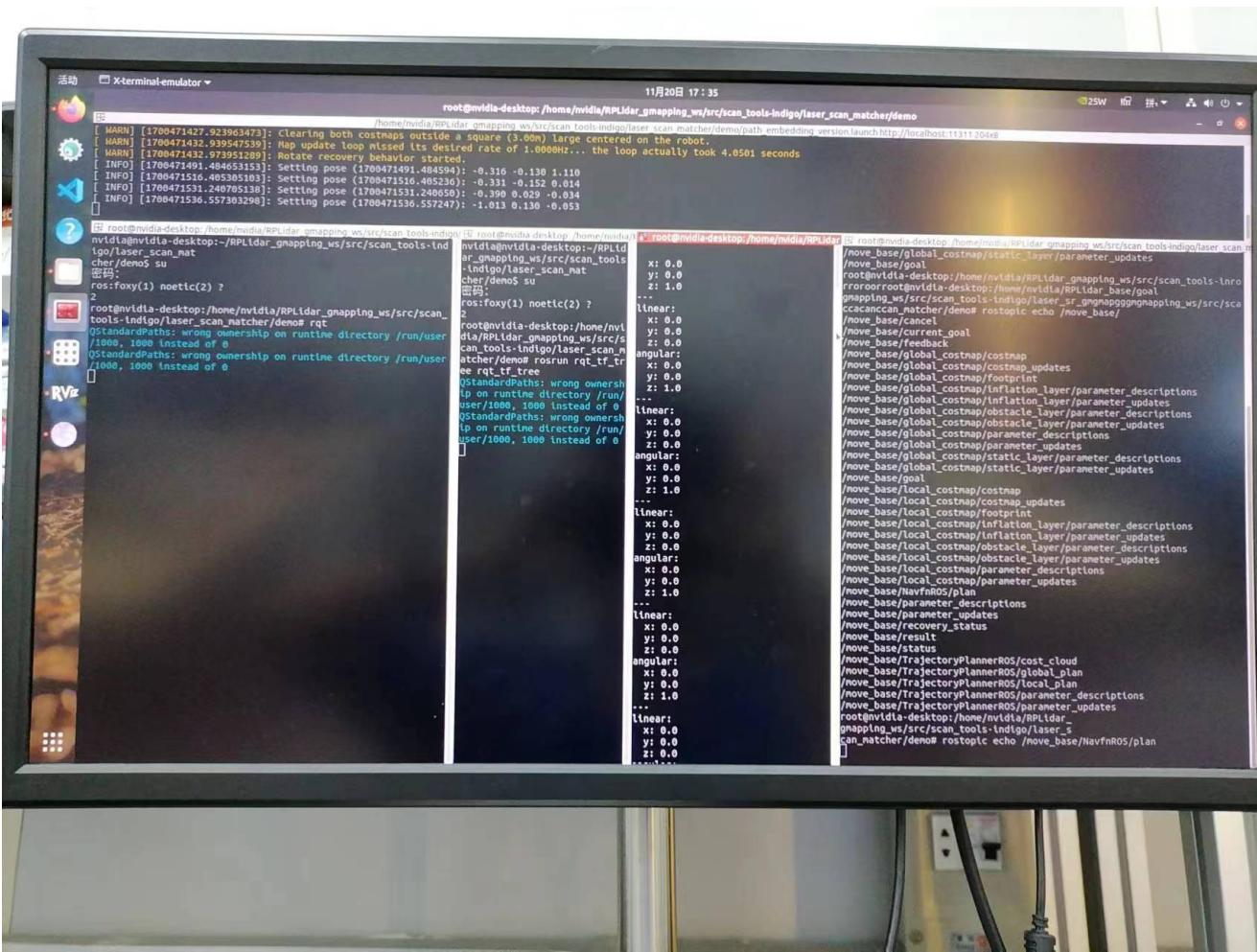
  <include file="/home/nvidia/RPLidar_gmapping_ws/src/rplidar_ros-
master (1)/rplidar_ros-master/launch/rplidar.launch"/>
  <include file="/home/nvidia/RPLidar_gmapping_ws/src/scan_tools-
indigo/laser_scan_matcher/demo/aamcl.launch"/>
  <include file="/home/nvidia/RPLidar_gmapping_ws/src/scan_tools-
indigo/laser_scan_matcher/demo/path.launch"/>

  <node pkg="rviz" type="rviz" name="rviz"/>
</launch>

```



途中为渲染了costmap和粒子滤波的地图效果图



图中为向下位机发送的速度信息

1.3.1 接受雷达/点云数据

尽管官网ros_wiki解释有多种传感器用于导航，但传统的ros navigation只提供了laser_scan 和 pointcloud两种数据接收格式在sensor_msgs当中

1, 发送端雷达数据：

参考gmapping文件

```

<param name="linearUpdate" value="1.0"/>
<param name="angularUpdate" value="0.5"/>
<param name="temporalUpdate" value="0.4"/>
<param name="resampleThreshold" value="0.5"/>
<param name="particles" value="10"/>
<param name="xmin" value="-5.0"/>
<param name="ymin" value="-5.0"/>
<param name="xmax" value="5.0"/>
<param name="ymax" value="5.0"/>
<param name="delta" value="0.02"/>
<param name="llsamplerange" value="0.01"/>
<param name="llsamplestep" value="0.05"/>
<param name="lasamplerange" value="0.05"/>
<param name="lasamplestep" value="0.05"/>
```

2, 接收端雷达数据:

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min #起始扫描角度(rad)
float32 angle_max #终止扫描角度(rad)
float32 angle_increment #测量值之间的角距离(rad)
float32 time_increment #测量间隔时间(s)
float32 scan_time #扫描间隔时间(s)
float32 range_min #最小有效距离值(m)
float32 range_max #最大有效距离值(m)
float32[] ranges #一个周期的扫描数据
float32[] intensities #扫描强度数据, 如果设备不支持强度数据, 该数组为空
```

1.3.2 参考笔记

[bandasaikrishna/Autonomous_Mobile_Robot: Autonomous mobile robot navigation using ROS Navigation Stack. \(github.com\)](https://github.com/bandasaikrishna/Autonomous_Mobile_Robot)

在同一launch文件下开启以下结点是完成导航的最低要求

```
<?xml version="1.0"?>
<launch>

  <arg name="map_file" default="$(find
mobile_robot_autonomous_navigation)/maps/layout.yaml"/>
  <arg name="model" default="$(find
mobile_robot_autonomous_navigation)/urdf/mobile_robot.urdf.xacro"/>

  <rosparam file="$(find
mobile_robot_autonomous_navigation)/config/controllers.yaml"
command="load"/>
  <rosparam file="$(find
mobile_robot_autonomous_navigation)/config/joint_limits.yaml"
command="load"/>

  <param name="robot_description" command="$(find xacro)/xacro.py $(arg
model)" />

  <node name="robot_hardware_interface"
pkg="mobile_robot_autonomous_navigation"
type="mobile_robot_hardware_interface" output="screen">
    <remap from="/mobile_robot/mobile_base_controller/cmd_vel"
to="/mobile_robot/mobile_base_controller/cmd_vel"/>
  </node>
```

```

        to="/cmd_vel"/>
    </node>

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

    <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
mobile_robot_autonomous_navigation)/config/nav_config.rviz"/>

    <node name="controller_spawner" pkg="controller_manager"
type="spawner" respawn="false" output="screen"
        args="
            /mobile_robot/joints_update
            /mobile_robot/mobile_base_controller" >
    </node>

<node name="map_server" pkg="map_server" type="map_server"
args="$(arg map_file)" >
    </node>

    <include file="$(find rplidar_ros)/launch/rplidar.launch" />
    <include file="$(find
mobile_robot_autonomous_navigation)/launch/amcl.launch" />
    <include file="$(find
mobile_robot_autonomous_navigation)/launch/move_base.launch" />

</launch>

```

其中

```

<arg name="map_file" default="$(find
mobile_robot_autonomous_navigation)/maps/layout.yaml"/>
    <node name="map_server" pkg="map_server" type="map_server" args="$(arg
map_file)" >
        </node>

```

是预制地图文件的导入

```

<include file="$(find rplidar_ros)/launch/rplidar.launch" />

```

是雷达驱动文件的启动

```
<include file="$(find  
mobile_robot_autonomous_navigation)/launch/move_base.launch" />
```

是路径规划的文件包，其中包括了全局的和局部的路径规划，分别放在不同的路径规划命名空间下

```
<?xml version="1.0"?>  
<launch>  
  
    <arg name="base_global_planner" default="navfn/NavfnROS"/>  
    <arg name="base_local_planner"  
        default="dwa_local_planner/DWAPlannerROS"/>  
  
    <node pkg="move_base" type="move_base" respawn="false"  
        name="move_base" output="screen">  
        <param name="base_global_planner" value="$(arg  
            base_global_planner)"/>  
        <param name="base_local_planner" value="$(arg  
            base_local_planner)"/>  
        <rosparam file="$(find  
            mobile_robot_autonomous_navigation)/config/planner.yaml"  
            command="load"/>  
  
        <rosparam file="$(find  
            mobile_robot_autonomous_navigation)/config/common_costmap.yaml"  
            command="load" ns="global_costmap" />  
        <rosparam file="$(find  
            mobile_robot_autonomous_navigation)/config/common_costmap.yaml"  
            command="load" ns="local_costmap" />  
  
        <rosparam file="$(find  
            mobile_robot_autonomous_navigation)/config/local_costmap.yaml"  
            command="load" ns="local_costmap" />  
        <rosparam file="$(find  
            mobile_robot_autonomous_navigation)/config/global_costmap.yaml"  
            command="load" ns="global_costmap" />  
    </node>  
</launch>
```

该方案中使用的是i2c通信协议，特地了解

也就是说，对应于计算机网络中的物理层的一种规定电气特性的协议，能够让不同芯片之间的通信方式尽可能简单的方法就是I2C

2三维导航

2.1二维导航系统到三维导航系统

2.1.1关于Ros2通信机制的改变说明

ros2和ros1的底层通信机制需要适当的了解，在导航任务这种大量的数据传递任务中底层一点的报错非常常见

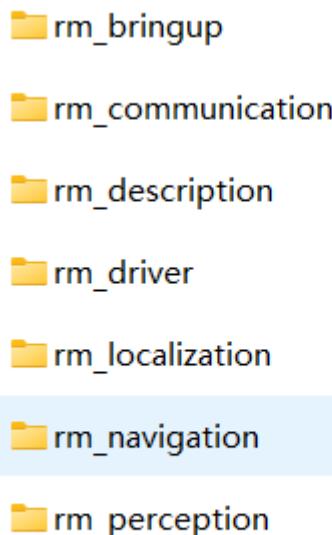
[翻译官小鱼带你看：史上最详细ROS2与ROS1对比，没有之一 - 知乎](#)

2.1.2 关于Ros2版本的选择对Nav2系统的影响

影响最大的肯定是[2.2.6 rm_navigation 导航任务调度层](#)，因为一旦版本改变，nav2对应的版本也会发生变化，这样支持调用的参数也会随之变化，同时要注意后面加入的传感器对Ros2系统的支持情况，选择尽可能满足需求并且代码简洁的版本是关键

2.2 三维导航系统的详细介绍

这里我将会详细介绍我们的导航框架，并简单介绍可能的未来的发展



2.2.1 rm_driver 驱动层

这里放置所用外设的驱动

特别需要关注的是雷达自身发送出来的消息类型，官方提供的两种消息类型对应在文件CustomMsg.msg和CustomPoint.msg中，在日后的开发中如果需要引入新的雷达数据处理算法一定要提前主要利用的数据类型，而不同的消息类型之间的兼容性是可以被处理的，但处理手法的好坏决定了日后整个导航系统的流畅程度，因为特别要注意结点的命名问题和qos问题，我们可以这样理解，跟大多闭源的系统相比，导航这套系统本身是开源的，也就是说我们时刻都会有数据输入，而对数据输入的鲁棒性决定了整个系统“源”流水是正常的

📎 需要关注的点

- 雷达数据的范围，通过echo测试(线束数是常见的雷达指标)

```
uint8 line           # laser number in lidar
```

- 雷达数据发送的频率，这在launch文件中是可调的，而接受数据的那端通常可调，收发的队列模型可以参考常见的qos问题

```
publish_freq = 10.0 # frequency of publish, 5.0, 10.0, 20.0, 50.0, etc.
```

- 雷达数据格式对其
- 发数据结点一致性，尽量只用一个总结点控制雷达的数据，并且做到在整个程序运行过程中不会中断

雷达的结点分别为点云的发送结点和imu结点，点云结点后续工作重点在点云数据融合imu结点的重点在数据的纠偏和去噪，在做回环的时候imu结点非常重要，须多做测试

```
void DriverNode::PointCloudDataPollThread()
{
    std::future_status status;
    std::this_thread::sleep_for(std::chrono::seconds(3));
    do {
        lddc_ptr_->DistributePointCloudData();
        status = future_.wait_for(std::chrono::microseconds(0));
    } while (status == std::future_status::timeout);
}

void DriverNode::ImuDataPollThread()
{
    std::future_status status;
    std::this_thread::sleep_for(std::chrono::seconds(3));
    do {
        lddc_ptr_->DistributeImuData();
    }
```

```
    status = future_.wait_for(std::chrono::microseconds(0));
} while (status == std::future_status::timeout);
}
```

2.2.2 rm_communication 通信层

这里放置所用队里的上下位机接口

用sentry_down，不过多解释，看懂下面的状态机相关的就行，其他的队里保持一致了

```
elif self.name == "sentry_down":
    self.chassis_sub = self.create_subscription(
        Twist, '/cmd_vel', self.ex_chassis_callback, 10)
```

以后统一在对应的 callback接口里更改发送内容，上下位机联调

```
def ex_chassis_callback(self, msg: Twist) -> None:
    """Chassis function, send chassis infomation to MCU.

    底盘功能，发送底盘信息给MCU

    Parameters
    -----
    msg: `Chassis`
        A chassis message is received
    """
    print("recived data chassis")

    self.robot_serial.send_data("chassis_ctrl", [msg.linear.x,
msg.linear.y, msg.linear.z, msg.angular.x, msg.angular.y,
msg.angular.z])
```

2.2.3 rm_description 机器人模型

这里放置urdf等机器人的关节变换文件,我这里只尝试了最基础的操作，跟二维导航一样，这里涉及到tf树的稳定性，必须尽量少改慎改

```

<joint name="base_footprint_joint" type="fixed">

  <origin xyz="0 0 0" rpy="0 0 0" />

  <parent link="base_footprint" />

  <child link="base_link" />

</joint>

```

2.2.4 rm_localization 定位层

导航任务中定位需要的功能写在这里,

pointlio可以说是导航最重要的结点，决定了整个导航系统能否建立，日后有问题首先排查这个包

一定要调整好的是结点的quality数据，记住雷达的数据源决定一切，要根据环境和需求调整结点队列

```

rclcpp::Subscription<sensor_msgs::msg::PointCloud2>::SharedPtr
sub_pcl_pc_;

rclcpp::Subscription<livox_ros_driver2::msg::CustomMsg>::SharedPtr
sub_pcl_livox_;

if (p_pre->lidar_type == AVIA){

    sub_pcl_livox_ = nh-
>create_subscription<livox_ros_driver2::msg::CustomMsg>(lid_topic, 20,
livox_pcl_cbk);

} else {

    sub_pcl_pc_ = nh-
>create_subscription<sensor_msgs::msg::PointCloud2>(lid_topic,
rclcpp::SensorDataQoS(),

standard_pcl_cbk);

}

auto sub_imu = nh->create_subscription<sensor_msgs::msg::Imu>
(imu_topic, 200, imu_cbk);

```

```

    auto pubLaserCloudFullRes = nh-
>create_publisher<sensor_msgs::msg::PointCloud2>

("cloud_registered", 100);

    auto pubLaserCloudFullRes_body = nh-
>create_publisher<sensor_msgs::msg::PointCloud2>

("cloud_registered_body", 100);

    auto pubLaserCloudEffect = nh-
>create_publisher<sensor_msgs::msg::PointCloud2>

("cloud_effected", 100);

    auto pubLaserCloudMap = nh-
>create_publisher<sensor_msgs::msg::PointCloud2>

("Laser_map", 100);

    auto pubOdomAftMapped = nh-
>create_publisher<nav_msgs::msg::Odometry>

("odom", 100);

    auto pubPath = nh->create_publisher<nav_msgs::msg::Path>

("path", 100);

//auto plane_pub = nh-
>create_publisher<visualization_msgs::msg::Marker>

//          ("/planner_normal", 1000);

    auto tf_broadcaster =
std::make_shared<tf2_ros::TransformBroadcaster>(nh);

```

2.2.5 rm_perception 传感器信号处理

segmentation结点需要注意的参数主要有

```

max_fit_error: 0.075      # maximum error of a point during line
fit.

long_threshold: 1.0        # distance between points after which

```

```
they are considered far from each other.
```

```
max_long_height: 0.1          # maximum height change to previous  
point in long line.
```

```
max_start_height: 0.2          # maximum difference to estimated  
ground height to start a new line.
```

```
line_search_angle: 0.2          # how far to search in angular  
direction to find a line [rad].
```

其中，需要对齐的结点在：

```
input_topic: "livox/lidar"  
  
obstacle_output_topic: "segmentation/obstacle"  
  
ground_output_topic: "segmentation/ground"
```

导航任务中对传感器信息的处理集中在这

pointcloud2scan中对齐的接口在

```
package='pointcloud_to_laserscan',  
executable='pointcloud_to_laserscan_node',  
remappings=[('cloud_in', ['/segmentation/obstacle']),  
           ('scan', ['/scan'])],
```

注意在segmentation中只要obstacle的部分，ground部分不要导入

其余参数记得要向雷达的处理结点和后续结点对齐

```
parameters=[{  
  
    'target_frame': 'livox_frame',  
  
    'transform_tolerance': 0.01,  
  
    'min_height': -3.0,  
  
    'max_height': 3.0,
```

2.2.6 rm_navigation 导航任务调度层

对nav2_params.yaml调试参数进行说明:这里最好是阅读nav2官网的数据手册来调试,

```
amcl:  
  
ros__parameters:  
  
    use_sim_time: False  
  
    alpha1: 0.2  
  
    alpha2: 0.2  
  
    alpha3: 0.2  
  
    alpha4: 0.2  
  
    alpha5: 0.2  
  
    base_frame_id: "base_footprint"  
  
    beam_skip_distance: 0.5  
  
    beam_skip_error_threshold: 0.9  
  
    beam_skip_threshold: 0.3  
  
    do_beamskip: false  
  
    global_frame_id: "map" #这种带id的要注意对齐开发  
  
    lambda_short: 0.1  
  
    laser_likelihood_max_dist: 2.0  
  
    laser_max_range: 100.0      #雷达的对应参数要适配调试  
  
    laser_min_range: -1.0  
  
    laser_model_type: "likelihood_field"  
  
    max_beams: 60  
  
    max_particles: 2000  
  
    min_particles: 500
```

```
odom_frame_id: "odom"

pf_err: 0.05

pf_z: 0.99

recovery_alpha_fast: 0.0

recovery_alpha_slow: 0.0

resample_interval: 1

robot_model_type: "differential"

#测试结果是全向轮和差速轮模式foxy的结果向差不大
#估计是后面升级其他版本就没有维护这个了

save_pose_rate: 0.5

sigma_hit: 0.2

tf_broadcast: true

transform_tolerance: 1.0

update_min_a: 0.2

update_min_d: 0.25

z_hit: 0.5

z_max: 0.05

z_rand: 0.5

z_short: 0.05

scan_topic: /scan      #这里后面可以尝试调试/PointCloud的，但现成的方
案很少

amcl_map_client:

ros__parameters:

use_sim_time: False
```

```

#如果以后考虑上模拟器的话可以开，但是一定要知道的是在这个调试
#一定会影响到tf树的转换，所以所有use_sim_time参数，包括其他包里的都要一致，
#具体可以看看ros2实现时间墙的具体机制

amcl_rclcpp_node:

ros__parameters:
    use_sim_time: False


bt_navigator:

ros__parameters:
    use_sim_time: False

    global_frame: map

    robot_base_frame: base_link

    odom_topic: /odom

    enable_groot_monitoring: True

    groot_zmq_publisher_port: 1666

    groot_zmq_server_port: 1667

    default_bt_xml_filename: "navigate_w_replanning_and_recovery.xml"

plugin_lib_names:

#增添的结点名称，最复杂的地方，如果后面有考虑二次开发要记得这些结点的先后
依赖关系
    #可以用rqt node来查看结点的数据流动情况，
    #注意看active和非active情况，有些结点受状态机约束并不会随时出现
    #加入下面结点时要记得结点开启的先后顺序，即使可以认为是同时的情况
    #有些结点foxy这个版本维护的不是很好，进程函数栈没有完全delete，直接kill进
程

    - nav2_compute_path_to_pose_action_bt_node

    - nav2_follow_path_action_bt_node

```

- nav2_back_up_action_bt_node
- nav2_spin_action_bt_node
- nav2_wait_action_bt_node
- nav2_clear_costmap_service_bt_node
- nav2_is_stuck_condition_bt_node
- nav2_goal_reached_condition_bt_node
- nav2_goal_updated_condition_bt_node
- nav2_initial_pose_received_condition_bt_node
- nav2_reinitialize_global_localization_service_bt_node
- nav2_rate_controller_bt_node
- nav2_distance_controller_bt_node
- nav2_speed_controller_bt_node
- nav2_truncate_path_action_bt_node
- nav2_goal_updater_node_bt_node
- nav2_recovery_node_bt_node
- nav2_pipeline_sequence_bt_node
- nav2_round_robin_node_bt_node
- nav2_transform_available_condition_bt_node
- nav2_time_expired_condition_bt_node
- nav2_distance_traveled_condition_bt_node

bt_navigator_rclcpp_node:

ros__parameters:

use_sim_time: False

```
controller_server:

ros__parameters:

    use_sim_time: False

    controller_frequency: 20.0

    min_x_velocity_threshold: 0.001

    min_y_velocity_threshold: 0.5

    min_theta_velocity_threshold: 0.001

    progress_checker_plugin: "progress_checker"

    goal_checker_plugin: "goal_checker"

    controller_plugins: ["FollowPath"]

# Progress checker parameters

progress_checker:

    plugin: "nav2_controller::SimpleProgressChecker"

    required_movement_radius: 0.5

    movement_time_allowance: 10.0

# Goal checker parameters

goal_checker:

    plugin: "nav2_controller::SimpleGoalChecker"

    xy_goal_tolerance: 0.25

    yaw_goal_tolerance: 0.25

    stateful: True
```

```
#目标满意度的残差，注意这里

# DWB parameters

FollowPath:

plugin: "dwb_core::DWBLocalPlanner"

debug_trajectory_details: True

min_vel_x: 0.0

min_vel_y: 0.0

max_vel_x: 0.26

max_vel_y: 0.0

max_vel_theta: 1.0

min_speed_xy: 0.0

max_speed_xy: 0.26

min_speed_theta: 0.0

# Add high threshold velocity for turtlebot 3 issue.

# https://github.com/ROBOTIS-GIT/turtlebot3\_simulations/issues/75
#有很多nav2源码处的issue要多看看，明白别人是怎么解决的

acc_lim_x: 2.5

acc_lim_y: 0.0

acc_lim_theta: 3.2

decel_lim_x: -2.5

decel_lim_y: 0.0

decel_lim_theta: -3.2

vx_samples: 20

vy_samples: 5
```

```
vtheta_samples: 20

sim_time: 1.7

linear_granularity: 0.05

angular_granularity: 0.025

transform_tolerance: 0.2

xy_goal_tolerance: 0.25

trans_stopped_velocity: 0.25

short_circuit_trajectory_evaluation: True

stateful: True

critics: ["RotateToGoal", "Oscillation", "BaseObstacle",
"GoalAlign", "PathAlign", "PathDist", "GoalDist"]

BaseObstacle.scale: 0.02

PathAlign.scale: 32.0

PathAlign.forward_point_distance: 0.1

GoalAlign.scale: 24.0

GoalAlign.forward_point_distance: 0.1

PathDist.scale: 32.0

GoalDist.scale: 24.0

RotateToGoal.scale: 32.0

RotateToGoal.slowing_factor: 5.0

RotateToGoal.lookahead_time: -1.0

controller_server_rclcpp_node:

ros__parameters:
```

```
use_sim_time: False

local_costmap:

local_costmap:

ros__parameters:

    update_frequency: 8.0

    publish_frequency: 8.0

    global_frame: odom

    robot_base_frame: base_link

    use_sim_time: False

    rolling_window: true

    width: 100

    height: 100

    resolution: 0.1

    robot_radius: 0.2

    plugins: ["voxel_layer", "inflation_layer"]

    inflation_layer:

        plugin: "nav2_costmap_2d::InflationLayer"

        cost_scaling_factor: 3.0

        inflation_radius: 0.3

    voxel_layer:

        plugin: "nav2_costmap_2d::VoxelLayer"

        enabled: True

        publish_voxel_map: True
```

```
origin_z: 0.0

z_resolution: 0.05

z_voxels: 16

max_obstacle_height: 2.0

mark_threshold: 0

#observation_sources: scan

observation_source: livox

#scan:

#  topic: /scan

#  max_obstacle_height: 2.0

#  clearing: True

#  marking: True

#  data_type: "LaserScan"

livox:

topic: /scan

#add

sensor_frame: livox_frame

max_obstacle_height: 2.0

clearing: True

marking: True

data_type: "LaserScan"

static_layer:

map_subscribe_transient_local: True
```

```
    always_send_full_costmap: True

local_costmap_client:

ros__parameters:
    use_sim_time: False

local_costmap_rclcpp_node:

ros__parameters:
    use_sim_time: False

global_costmap:

global_costmap:
    ros__parameters:
        update_frequency: 8.0
        publish_frequency: 8.0
        global_frame: map
        robot_base_frame: base_link
        use_sim_time: False
        robot_radius: 0.3
        resolution: 0.1
        track_unknown_space: true
    plugins: ["static_layer", "obstacle_layer", "inflation_layer"]

obstacle_layer:
    plugin: "nav2_costmap_2d::ObstacleLayer"
    enabled: True
    observation_sources: scan
```

```
scan:

    topic: /scan

    #add

    sensor_frame: livox_frame

    max_obstacle_height: 2.0

    clearing: True

    marking: True

    data_type: "LaserScan"

#这里后面可以尝试自定义数据的输入类型，要结合源码一起看了

#scan:

#    topic: /scan

#    max_obstacle_height: 2.0

#    clearing: True

#    marking: True

#    data_type: "LaserScan"

static_layer:

    plugin: "nav2_costmap_2d::StaticLayer"

    map_subscribe_transient_local: True

inflation_layer:

    plugin: "nav2_costmap_2d::InflationLayer"

    cost_scaling_factor: 3.0

    inflation_radius: 0.1

    always_send_full_costmap: True

global_costmap_client:
```

```
ros__parameters:
    use_sim_time: False

global_costmap_rclcpp_node:
    ros__parameters:
        use_sim_time: False

map_server:
    ros__parameters:
        use_sim_time: False
        yaml_filename: "test.yaml"

map_saver:
    ros__parameters:
        use_sim_time: False
        save_map_timeout: 5000
        free_thresh_default: 0.25
        occupied_thresh_default: 0.65
        map_subscribe_transient_local: False

planner_server:
    ros__parameters:
        expected_planner_frequency: 20.0
        use_sim_time: False
```

```
planner_plugins: ["GridBased"]

GridBased:

    plugin: "nav2_navfn_planner/NavfnPlanner"

    tolerance: 0.5

    use_astar: false

    allow_unknown: true

planner_server_rclcpp_node:

ros__parameters:

    use_sim_time: False

recoveries_server:

ros__parameters:

    costmap_topic: local_costmap/costmap_raw

    footprint_topic: local_costmap/published_footprint

    cycle_frequency: 10.0

    recovery_plugins: ["spin", "back_up", "wait"]

spin:

    plugin: "nav2_recoveries/Spin"

back_up:

    plugin: "nav2_recoveries/BackUp"

wait:

    plugin: "nav2_recoveries/Wait"

global_frame: odom
```

```
robot_base_frame: base_link

transform_timeout: 0.1

use_sim_time: False

simulate_ahead_time: 2.0

max_rotational_vel: 1.0

min_rotational_vel: 0.4

rotational_acc_lim: 3.2

robot_state_publisher:

ros__parameters:

use_sim_time: False
```

调用nav2的输入参数都放在这，曾经尝试过直接调用源码来作这一层，但是实践证明调用系统的往往效果更好，代码更加简单明了

对mapper_params_online_async.yaml调试参数进行说明：

```
slam_toolbox:

ros__parameters:

# Plugin params

solver_plugin: solver_plugins::CeresSolver

ceres_linear_solver: SPARSE_NORMAL_CHOLESKY

ceres_preconditioner: SCHUR_JACOBI

ceres_trust_strategy: LEVENBERG_MARQUARDT

ceres_dogleg_type: TRADITIONAL_DOGLEG
```

```
ceres_loss_function: None

# ROS Parameters

odom_frame: odom

map_frame: map

base_frame: base_footprint

scan_topic: /scan

mode: mapping #localization

# if you'd like to immediately start continuing a map at a given
pose

# or at the dock, but they are mutually exclusive, if pose is given
# will use pose

#map_file_name: test_steve

# map_start_pose: [0.0, 0.0, 0.0]

#map_start_at_dock: true

debug_logging: false

throttle_scans: 1

transform_publish_period: 0.02 #if 0 never publishes odometry

map_update_interval: 5.0

resolution: 0.05

max_laser_range: 20.0 #for rastering images

minimum_time_interval: 0.5
```

```
transform_timeout: 0.2

tf_buffer_duration: 30.

stack_size_to_use: 40000000 // program needs a larger stack size
to serialize large maps

enable_interactive_mode: true

# General Parameters

use_scan_matching: true

use_scan_barycenter: true

minimum_travel_distance: 0.5

minimum_travel_heading: 0.5

scan_buffer_size: 10

scan_buffer_maximum_scan_distance: 10.0

link_match_minimum_response_fine: 0.1

link_scan_maximum_distance: 1.5

loop_search_maximum_distance: 3.0

do_loop_closing: true

loop_match_minimum_chain_size: 10

loop_match_maximum_variance_coarse: 3.0

loop_match_minimum_response_coarse: 0.35

loop_match_minimum_response_fine: 0.45

# Correlation Parameters - Correlation Parameters

correlation_search_space_dimension: 0.5
```

```
correlation_search_space_resolution: 0.01

correlation_search_space_smear_deviation: 0.1

# Correlation Parameters - Loop Closure Parameters

loop_search_space_dimension: 8.0

loop_search_space_resolution: 0.05

loop_search_space_smear_deviation: 0.03

# Scan Matcher Parameters

distance_variance_penalty: 0.5

angle_variance_penalty: 1.0

fine_search_angle_offset: 0.00349

coarse_search_angle_offset: 0.349

coarse_angle_resolution: 0.0349

minimum_angle_penalty: 0.9

minimum_distance_penalty: 0.5

use_response_expansion: true
```

2.3 参考网站

[Ubuntu 20.04使用Livox Mid-360_livox mid360 ubuntu20-CSDN博客](#)

[【PCL build issue】WARNING io features related to pcap_png will be disabled_warning ** io features related to pcap will be -CSDN博客](#)

[baiyeweiguang/CSU-RM-Sentry: 中南大学FYT战队RM哨兵机器人上位机算法](#)

[PCL智能指针Ptr详解-CSDN博客](#)

[Remove ROS1 headers from point_cloud.hpp \(#410\) · ros-perception/perception_pcl@3bb07c0](#)

[LihanChen2004/pcd2pgm: Based on ROS2 and PCL libraries, used to convert pcd point cloud files into raster maps for navigation](#)

[使用实体Turtlebot 3导航 — Navigation 2 1.0.0 文档](#)

[求助pointcloud_to_laserscan转换后rviz无法显示问题 | 鱼香ROS](#)

[《动手学ROS2》10.7 Nav2导航框架介绍与安装_鱼香ROS的博客-CSDN博客](#)

[鱼香ROS-CSDN博客](#)

[《动手学ROS2》10.9使用FishBot进行自主导航-CSDN博客](#)

[AMCL \(自适应蒙特卡洛定位\) — Navigation 2 1.0.0 文档](#)

[ros基础必看之各个frame的理解 ros frame-CSDN博客](#)

[Linux/Ubuntu安装图形界面版clash-gui | MGodmonkeyの世界](#)

[【已解决】Ubuntu20.04安装fmt编译报错：error: no matching function for call to 'fmt::v11::formatter的解决方法_error: no matching function for call to 鈥櫔mt::v11:-CSDN博客](#)

[Releases · fmtlib/fmt](#)

[使用mid360从0开始搭建实物机器人入门级导航系统，基于FastLio,Move_Base业界新闻_筋斗云](#)

[配置激活map_server报错，Transitioning failed | 鱼香ROS](#)

[ros2中rviz无地图显示frame\[map\]does not exist | 鱼香ROS](#)

[Navigation 2系列教程（六）——普通教程之四：使用SLAM制图的同时进行导航 - 知乎](#)

[GitCode - 全球开发者的开源社区,开源代码托管平台](#)

[带你理清：ROS机器人导航功能实现、解析、以及参数说明_ros中如何指定机器人去哪个位置-CSDN博客](#)

[Can't update static costmap layer, no map received. Connecting to the GO2 Robot Via cyclonedds · Issue #87 · abizovnuralem/go2_ros2_sdk](#)

[local costmap,_globalmap no map received. · Issue #4288 · ros-navigation/navigation2](#)

[Local costmap not displayed \(Could not transform from \[odom\] to \[map\]\) · Issue #921 · ros-navigation/navigation2](#)

[navigation2/nav2_bringup/bringup at foxy-devel · ros-navigation/navigation2](#)

[Re-enable use of tf message filter by shiveshkhaitan · Pull Request #375 · ros2/rviz](#)

[rviz2中不显示代价地图 | 鱼香ROS](#)

[ros2 - No map received warning when launching nav2 with rviz2 using custom robot simulation on gazebo - Stack Overflow](#)

[navigation - Where is max_obstacle_height in an Obstacle_Layer measured from? - Robotics Stack Exchange](#)

[TEB 调整指引 | docs.movel.ai](#)

[ros - How to use "pointcloud_to_laserscan"? - Robotics Stack Exchange](#)

[无人驾驶\(四\)---远程桌面控制工具: NoMachine踩坑记录_nomachine远程桌面报108-CSDN博客](#)

[Nav2 Navigation System](#)

[Robots/TiAGo/Tutorials/motions/cmd_vel - ROS Wiki](#)

[ROS学习二：cmd_vel_cmdvel-CSDN博客](#)

[Resizing Costmap · Issue #2177 · ros-navigation/navigation2](#)

3 附录

3.1 关于数理知识

关于数学物理部分网上有很多的参考资料，同时大量的论文是默认理解并熟悉的，所以为了后续的开发和完善，这里将我认为的部分重点知识简单的概括一遍，但本人也不是特别擅长繁杂的公式推理，只能起到点到即止的作用，不足之处只能反复迭代完善了，同时为了方便对接常用的术语和论文的表述，这里会用英语进行叙述

3.1.1 李群李代数与刚体运动

1 Using Matrix to describe rotation



Note

every vector only refer to one asymmetry matrix, you can use both of them to represent a rotation

- there is a mathematics trick that by giving expanding dimension with the value 1 can we obtain a better equation form contain both rotation and pretranslation

$$\begin{bmatrix} a' \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} a \\ 1 \end{bmatrix}$$

- to make things less-redundancy we can use the eigenvector to unique refer to the transformation matrix
 - euler angle is another suggested method to describe such a thing(despite the existance of Gimbal Lock Question)
- 2Using Quaternion to describe rotation
- the foundamental theory of Quaternion:

$$i^2 = j^2 = k^2 = ijk = -1$$

- how can we describe roratain by Quaternion?
- suggest we describe the rotation by vector v and the angle theta

$$v(v1, v2, v3)$$

$$\theta$$

$$w = a + xi + yj + zk$$

$$\begin{cases} a = \cos(\theta/2) \\ x = v1 * \sin(\theta/2) \\ y = v2 * \sin(\theta/2) \\ z = v3 * \sin(\theta/2) \end{cases}$$

the rotation equation goes with:

$$p' = wpw^{-1}$$

3Clibrary to describe matrix:eigen

Note

it is very simple to import eigen in cmakelists

```
include_directories("/usr/include/eigen3")
```

- some basic task done by eigen
- [Eigen/Matlab库矩阵运算方法_泸州月的博客-CSDN博客](#)

```

#include <iostream>

using namespace std;

#include <ctime>
#include <Eigen/Core>
#include <Eigen/Dense>

using namespace Eigen;

#define MATRIX_SIZE 50

int main(){
    Matrix<float,2,3> matrix_23;
    Vector3d v_3d;
    Matrix<float,3,1> vd_3d;//Too few coefficients passed to comma
initializer (operator<<)
                //was reported as a bug when you input too
few argument in the varieble    Matrix3d matrix_33 = Matrix3d::Zero();

    Matrix<double,Dynamic,Dynamic>matrix_dynamic;

    matrix_23 << 1 ,2, 3, 4, 5, 6;

    cout<<"matrix 2x3 from 1 to 6:\n"<<matrix_23<<endl;

    cout<<"print matrix 2x3\n";

    for(int i = 0;i< 2;i++){
        for(int j = 0;j < 3;j++){
            cout<<matrix_23(i,j)<<"\t";
        }
    }
    cout<<endl;

    v_3d << 3, 2, 1;
    vd_3d << 4, 5, 6;

    Matrix<double,2,1>result = matrix_23.cast<double>() * v_3d;

    cout<<"[1,2,3;4,5,6]*[4,5,6]: "<<result.transpose() << endl;

    matrix_33 = Matrix3d::Random();
}

```

```

cout<<"random matrix: \n"<<matrix_33<<endl;

cout<<"transpose: \n"<<matrix_33.transpose()<<endl;

cout<<"sum: \n"<<matrix_33.sum()<<endl;

cout<<"trace: \n"<<matrix_33.trace()<<endl;

cout<<"times 10: \n"<<10 * matrix_33<<endl;

cout<<"inverse: \n"<<matrix_33.inverse()<<endl;

cout<<"det: \n"<<matrix_33.determinant()<<endl;

SelfAdjointEigenSolver<Matrix3d> eigen_solver(matrix_33.transpose()
* matrix_33);

cout<<"Eigen value = \n" <<eigen_solver.eigenvalues()<<endl;

cout<<"Eigen vectors = \n" <<eigen_solver.eigenvectors()<<endl;

return 0;
}

```

- strp further, there is eigengeometry
- you can see the Quaternion in eigengeometry
-

```

#include <iostream>
#include <cmath>

using namespace std;

#include <eigen3/Eigen/Core>
#include <eigen3/Eigen/Geometry>

using namespace Eigen;

int main(int argc, char **argv){

    Matrix3d rotation_matrix = Matrix3d::Identity();

    AngleAxisd rotation_vector(M_PI / 4, Vector3d(0, 0, 1));

```

```

cout.precision(3);

cout<<"rotation_matrix = \n"<< rotation_vector.matrix() <<endl;

rotation_matrix = rotation_vector.toRotationMatrix();

Vector3d v(1,0,0);

Vector3d v_rotated = rotation_vector *v;

cout<<"(1,0,0) after rotation (by angle axis) = "
<<v_rotated.transpose()<<endl;

v_rotated = rotation_matrix*v;

cout<<"(1,0,0) after rotation (by matrix) = "<<v_rotated.transpose()
<<endl;

Vector3d euler_angles = rotation_matrix.eulerAngles(2,1,0);

cout<<"yaw pitch roll = "<<euler_angles.transpose()<<endl;

Isometry3d T = Isometry3d::Identity();

T.rotate(rotation_vector);

T.pretranslate(Vector3d(1,3,4));

cout<<"Transform matrix = \n"<<T.matrix()<<endl;

Vector3d v_transformed = T*v;

cout<<"v_transformed = "<<v_transformed.transpose()<<endl;

Quaterniond q = Quaterniond(rotation_vector);

cout<<"Quaterniond from rotation vector = "<<q.coeffs().transpose()
<<endl;

q = Quaterniond(rotation_matrix);

cout<<"Quaterniond from rotation matrix = "<<q.coeffs().transpose()
<<endl;

v_rotated = q*v;

cout<<"(1,0,0) after rotation = "<<v_rotated.transpose()<<endl;

```

```

cout<<"should be equal to "<<(q*Quaterniond(0,1,0,0) *
q.inverse().coeffs().transpose()<<endl;
    return 0;
}

```

- ofcourse you can transform Quaterniond into the normal rotataion vector

```

#include <iostream>
#include <vector>
#include <algorithm>
#include <eigen3/Eigen/Core>
#include <eigen3/Eigen/Geometry>

using namespace std;
using namespace Eigen;

int main(int argc,char ** argv){
    Quaterniond q1(0.35, 0.2, 0.3, 0.1), q2(-0.5, 0.4, -0.1, 0.2);
    q1.normalize();
    q2.normalize();
    Vector3d t1(0.3,0.1,0.1), t2(-0.1,0.5,0.3);
    Vector3d p1(0.5,0,0.2);

    Isometry3d T1w(q1), T2w(q2);
    T1w.pretranslate(t1);
    T2w.pretranslate(t2);

    Vector3d p2 = T2w*T1w.inverse() * p1;
    cout<<endl<<p2.transpose()<<endl;
    return 0;
}

```

display output is: -0.0309731 0.73499 0.296108

- use trajectory plot to display
 - cmakelists

```

cmake_minimum_required(VERSION 3.2)

project(plotTrajectory)

set(CMAKE_CXX_FLAGS "-std=c++14")

find_package(Pangolin REQUIRED)

```

```

set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)

aux_source_directory(src SRC_LIST)

include_directories(include)

include_directories("/usr/include/eigen3")

add_executable(plotTrajectory ${SRC_LIST})

target_link_libraries(plotTrajectory ${Pangolin_LIBRARIES})

```

- • cpp

```

#include <pangolin/pangolin.h>
#include <eigen3/Eigen/Core>
#include <unistd.h>

using namespace std;
using namespace Eigen;

string trajectory_file = "/home/vito/下载/trajectory.txt";

void DrawTrajectory(vector<Isometry3d,
Eigen::aligned_allocator<Isometry3d>>);

int main(int argc,char **argv){
    vector<Isometry3d, Eigen::aligned_allocator<Isometry3d>> poses;

    ifstream fin(trajectory_file);

    if(!fin){
        cout<<" cannot find trajectory file at "
<<trajectory_file<<endl;
        return 1;
    }

    while(!fin.eof()){
        double time, tx, ty, tz, qx, qy, qz, qw;

        fin>>time>>tx>>ty>>tz>>qx>>qy>>qz>>qw;

        Isometry3d Twr(Quaterniond(qw,qx,qy,qz));
        Twr.pretranslate(Vector3d(tx, ty, tz));
    }
}

```

```

        poses.push_back(Twr);

    }

cout<<"read total "<<poses.size()<<"poses entries"<<endl;

DrawTrajectory(poses);

return 0;
}

void DrawTrajectory(vector<Eigen::Isometry3d,
Eigen::aligned_allocator<Isometry3d>> poses){

pangolin::CreateWindowAndBind("trajectory View",1024,768);

glEnable(GL_DEPTH_TEST);

glEnable(GL_BLEND);

glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);

pangolin::OpenGLRenderState s_cam(
    pangolin::ProjectionMatrix(1024,768,500,500,512,389,0.1,1000),
    pangolin::ModelViewLookAt(0,-0.1,-1.8,0,0,0,0.0,-1.0,0.0)
);

pangolin::View &d_cam = pangolin::CreateDisplay()
    .SetBounds(0.0,1.0,0.0,1.0,-1024.0f/768.0f)
    .SetHandler(new pangolin::Handler3D(s_cam));

while(pangolin::ShouldQuit()== false){

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    d_cam.Activate(s_cam);
    glColor(1.0f,1.0f,1.0f,1.0f);
    glLineWidth(2);

    for(size_t i = 0;i<poses.size();i++){

        Vector3d Ow = poses[i].translation();
        Vector3d Xw = poses[i]*(0.1*Vector3d(1,0,0));
        Vector3d Yw = poses[i]*(0,1*Vector3d(0,1,0));
        Vector3d Zw = poses[i]*(0.1*Vector3d(0,0,1));
    }
}

```

```

glBegin(GL_LINES);

	glColor3f(1.0,0.0,0.0);
	glVertex3d(Ow[0],Ow[1],Ow[2]);
	glVertex3d(Xw[0],Xw[1],Xw[2]);

	glColor3f(0.0,1.0,0.0);
	glVertex3d(Ow[0],Ow[1],Ow[2]);
	glVertex3d(Yw[0],Yw[1],Yw[2]);

	glColor3f(0.0,0.0,1.0);
	glVertex3d(Ow[0],Ow[1],Ow[2]);
	glVertex3d(Zw[0],Zw[1],Zw[2]);

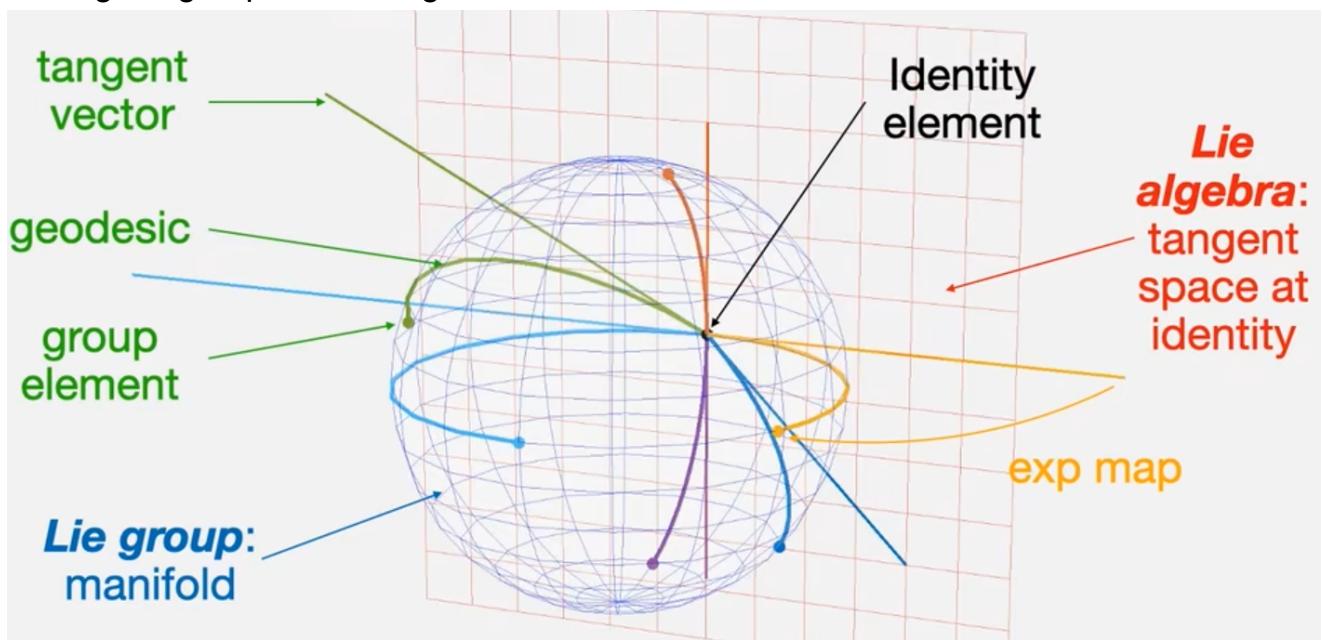
	glEnd();
}

for(size_t i = 0;i<poses.size();i++){
    glColor3f(0.0,0.0,0.0);
    glBegin(GL_LINES);
    auto p1 = poses[i], p2 = poses[i+1];
    glVertex3d(p1.translation()[0],p1.translation()
[1],p1.translation()[2]);
    glVertex3d(p2.translation()[0],p2.translation()
[1],p2.translation()[2]);
    glEnd();
}

pangolin::FinishFrame();
usleep(5000);
}

```

4 Using Lie group and Lie algebra to describe rotation

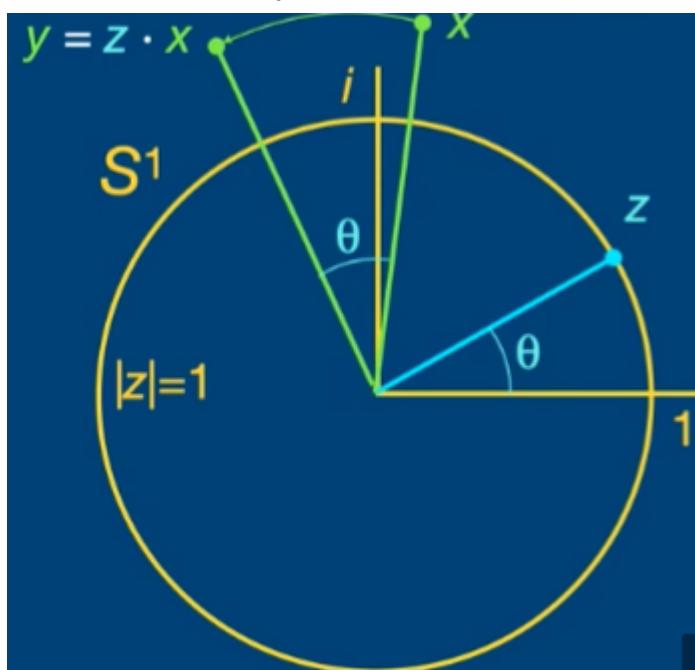


Note

we use group because they have the power to transform other elements

Basic knowledge of Lie group

- before simply list the basic component of lie group, we can firstly learn how to represent rotation by the complex number
- we know that by the following function can we represent a simple rotation in two dimension, namely a flat plane:



$$B = zA$$

$$z^*z = 1; ||z|| = 1$$

the sample can be proved:

-to prove B's length equals to A's length:(to get it simple, we use the notion of length)

$$A = X + Yi$$

$$z = \cos\theta + \sin\theta i$$

$$B = X\cos\theta + X\sin\theta i + Y\cos\theta i - Y\sin\theta$$

- you can simply get the equation below:

$$A^2 = X^2 + Y^2 = B^2$$

then the next part will be the proof of angle equation:

we use w to represent the angle of B and phi for A

$$\cos\omega = \cos(\theta + \phi) = \cos\theta\cos\phi - \sin\phi\sin\theta = \frac{X\cos\theta - Y\sin\theta}{\sqrt{X^2 + Y^2}}$$

you can prove the sin thing in the same way, remember the only combination of sin and cos can precisely choose an angle

you can expand the theory to SO(2) space:

$$R = I\cos\theta + [1]_x\sin\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

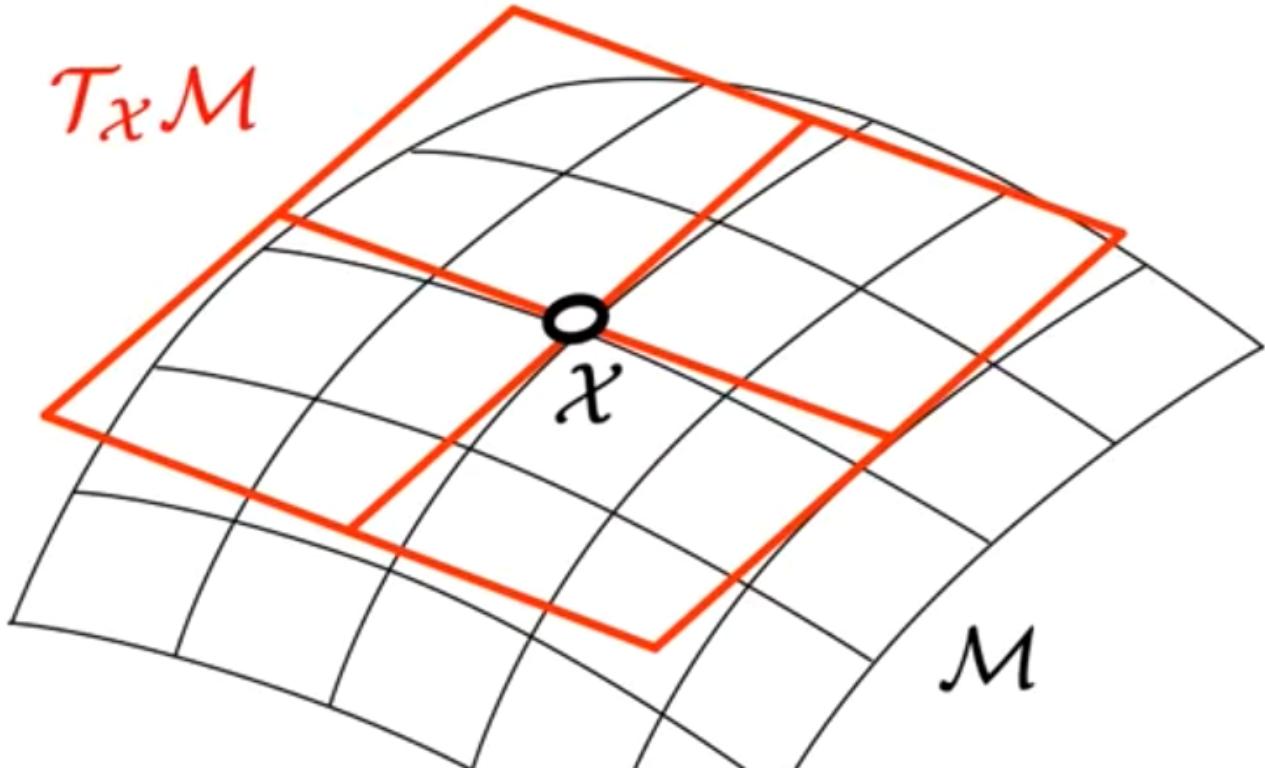
you will surprisingly find that we have something in common in the outcome of multiplying

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{pmatrix}$$

now we can really step into the space of SO(3):

Definition

a group is also a smooth manifold



Note

any Lie group have the Lie algebra accordingly, which used to describe the partial quality of Lie group

using sophuslib to describe Lie's elements

```

cmake_minimum_required(VERSION 3.2)

project(jointMap)

set(CMAKE_CXX_FLAGS "-std=c++14")

find_package(OpenCV REQUIRED)

find_package(Pangolin REQUIRED)

find_package(Sophus REQUIRED)

set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)

include_directories(include)

include_directories("/usr/include/eigen3")

```

```

include_directories(${OpenCV_INCLUDE_DIRS})

include_directories(${Sophus_INCLUDE_DIRS})

aux_source_directory(src SRC_LIST)

add_executable(jointMap ${SRC_LIST})

target_link_libraries(jointMap ${OpenCV_LIBS} ${Pangolin_LIBRARIES})

target_link_libraries(jointMap ${Sophus_LIBRARIES} fmt)

```

Note

注：sophus库的使用需要以下语句：

```

find_package(Sophus REQUIRED)
include_directories(${Sophus_INCLUDE_DIRS})
target_link_libraries(jointMap ${Sophus_LIBRARIES} fmt)

```

- cpp sophus的使用

```

#include <iostream>
#include <cmath>
#include <eigen3/Eigen/Core>
#include <eigen3/Eigen/Geometry>
#include "sophus/se3.hpp"

using namespace std;
using namespace Eigen;

int main(int argc, char **argv){
    Matrix3d R = AngleAxisd(M_PI / 2, Vector3d(0, 0, 1)).toRotationMatrix();

    Quaterniond q(R);

    Sophus::SO3d S03_R(R);
    Sophus::SO3d S03_q(q);

    cout<<"SO(3) from matrix:\n"<<S03_R.matrix()<<endl;
    cout<<"SO(3) from Quaternion:\n"<<S03_q.matrix()<<endl;

```

```

cout<<"they are equal"<<endl;

Vector3d so3 = S03_R.log();
cout<<"so3 = "<<so3.transpose()<<endl;

cout<<"so3 hat=\n"<<Sophus::S03d::hat(so3)<<endl;

cout<<"so3 hat vee = "<<Sophus::S03d::vee(Sophus::S03d::hat(so3))
<<endl;

Vector3d update_so3(1e-3, 0, 0);

Sophus::S03d S03_updated = Sophus::S03d::exp(update_so3) * S03_R;

cout<<"S03_updated = \n"<<S03_updated.matrix() <<endl;

cout<<"*****"*<<endl;

Vector3d t(1, 0, 0);

Sophus::SE3d SE3_Rt(R, t);

Sophus::SE3d SE3_qt(q, t);

cout<<" SE3 from R, t = \n"<<SE3_Rt.matrix()<<endl;

cout<<" SE3 from q, t = \n"<<SE3_qt.matrix()<<endl;

typedef Eigen::Matrix<double, 6, 1>Vector6d;

Vector6d se3 =SE3_Rt.log();

cout<<"se3 = "<<se3.transpose()<<endl;

cout<<"se3 hat = \n"<<Sophus::SE3d::hat(se3)<<endl;

cout<<"se3 hat vee= \n"
<<Sophus::SE3d::vee(Sophus::SE3d::hat(se3)).transpose()<<endl;

Vector6d update_se3;

update_se3.setZero();

update_se3(0, 0) = 1e-4d;

Sophus::SE3d SE3_updated = Sophus::SE3d::exp(update_se3) * SE3_Rt;

```

```

    cout<<"SE3_updated = "<<endl<<SE3_updated.matrix()<<endl;

    return 0;
}

```

- cpp 点云的绘制

```

#include<iostream>
#include<fstream>
#include<opencv4/opencv2/opencv.hpp>
#include<boost/format.hpp>
#include<pangolin/pangolin.h>
#include<sophus/se3.hpp>

using namespace std;

typedef vector<Sophus::SE3d,Eigen::aligned_allocator<Sophus::SE3d>>
TrajectoryType;
typedef Eigen::Matrix<double,6,1> Vector6d;

void showPointCloud(
    const
vector<Vector6d,Eigen::aligned_allocator<Vector6d>>&pointCloud
);

int main(int argc, char **argv){
    vector<cv::Mat> colorImgs, depthImgs;
    TrajectoryType poses;

    ifstream fin("./pose.txt");
    if(!fin){
        cerr<<"please run the program in file within pose.txt"<<endl;
        return 1;
    }

    for(int i = 0;i<5;i++){
        boost::format fmt("./%s/%d.%s");
        colorImgs.push_back(cv::imread((fmt % "color" %(i+1) %
"png").str()));
        depthImgs.push_back(cv::imread((fmt % "depth" %(i+1) %
"pgm").str(),-1));
    }

    double data[7] = {0};
    for(auto &d:data)
        fin>>d;
    Sophus::SE3d

```

```

pose(Eigen::Quaterniond(data[6],data[3],data[4],data[5]),
      Eigen::Vector3d(data[0],data[1],data[2]));
poses.push_back(pose);

}

double cx = 325.5;
double cy = 253.5;
double fx = 518.0;
double fy = 519.0;
double depthScale = 1000.0;
vector<Vector6d,Eigen::aligned_allocator<Vector6d>>pointCloud;

pointCloud.reserve(10000000);

for(int i = 0;i<5;i++){
    cout<<"switching the image"<<i+1<<endl;
    cv::Mat color = colorImg[i];
    cv::Mat depth = depthImg[i];
    Sophus::SE3d T = poses[i];
    for(int v = 0;v<color.rows;v++)
        for(int u = 0;u<color.cols;u++){
            unsigned int d = depth.ptr<unsigned short>(v)[u];
            if(d == 0) continue;
            Eigen::Vector3d point;

            point[2] = double(d)/depthScale;
            point[0] = (u - cx) * point[2] /fx;
            point[1] = (v - cy) * point[2] /fy;
            Eigen::Vector3d pointWorld = T*point;

            Vector6d p;
            p.head<3>() = pointWorld;
            p[5] = color.data[v * color.step + u *
color.channels()];
            p[4] = color.data[v * color.step + u *
color.channels()+1];
            p[3] = color.data[v * color.step + u *
color.channels()+2];
            pointCloud.push_back(p);

        }
    }

cout<<"the number of pointCloud is:"<<pointCloud.size()<<endl;
showPointCloud(pointCloud);
return 0;

```

```

}

void showPointCloud(const vector<Vector6d>
Eigen::aligned_allocator<Vector6d>>& pointCloud){

    if(pointCloud.empty()){
        cerr <<"Point cloud is empty!"<<endl;
        return;
    }

    pangolin::CreateWindowAndBind("Point Cloud Viewer",1024,768);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);

    pangolin::OpenGLRenderState s_cam(
        pangolin::ProjectionMatrix(1024,768,500,500,512,389,0.1,1000),
        pangolin::ModelViewLookAt(0,-0.1,-1.8,0,0,0,0.0,-1.0,0.0)
    );

    pangolin::View &d_cam = pangolin::CreateDisplay()
        .SetBounds(0.0,1.0,pangolin::Attach::Pix(175),1.0,-1024.0f/768.0f)
        .SetHandler(new pangolin::Handler3D(s_cam));

    while(pangolin::ShouldQuit()== false){
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        d_cam.Activate(s_cam);
        glColor3f(1.0f,1.0f,1.0f,1.0f);

        glPointSize(2);
        glBegin(GL_POINTS);
        for(auto &p: pointCloud){
            glColor3d(p[3]/255.0,p[4]/255.0,p[5]/255.0);
            glVertex3d(p[0],p[1],p[2]);
        }
        glEnd();
        pangolin::FinishFrame();
        usleep(5000);
    }
    return;
}

```

3.1.2 非线性优化

一阶二阶梯度法，

是以下两种现行优化方法的基础，所有优化方法都关注如何把梯度下降变得更加高效，这分为两个方面，量方面讨论了步长上的优化，可以减少总的迭代次数，质方面讨论如何利用二阶梯度来优化单次下降的近似程度

利用ceres解决一阶二阶优化问题

```
cmake_minimum_required(VERSION 3.2)
project(ceresCurveFitting)

set(CMAKE_BUILD_TYPE Release)
set(CMAKE_CXX_FLAGS "-std=c++14")

find_package(OpenCV REQUIRED)
find_package(Ceres REQUIRED)

set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)

aux_source_directory(src SRC_LIST)

include_directories(${OpenCV_INCLUDE_DIRS})
include_directories(${CERES_INCLUDE_DIRS})

add_executable(ceresCurveFitting ${SRC_LIST})
target_link_libraries(ceresCurveFitting ${OpenCV_LIBS}
${CERES_LIBRARIES})
```

Note

注：这里opencv只是用于产生基本的随机数利于优化；而ceres库的使用需要以下语句：

```
find_package(Ceres REQUIRED)
include_directories(${CERES_INCLUDE_DIRS})
target_link_libraries(ceresCurveFitting ${OpenCV_LIBS}
${CERES_LIBRARIES})
```

- 首先构造operator操作符

```

#include <iostream>
#include <opencv2/core/core.hpp>
#include <ceres/ceres.h>
#include <chrono>

using namespace std;

struct CURVE_FITTING_COST {
    CURVE_FITTING_COST(double x,double y) : _x(x), _y(y) {}

    template<typename T>
    bool operator()(const T * const abc, T *residual) const{
        residual[0] = T(_y) -
        ceres::exp(abc[0] * T(_x) * T(_x) + abc[1] * T(_x) +
        abc[2]);
        return true;
    }

    const double _x, _y;
};


```

- 构造曲线，其实是由点组成的vector数组

```

int main(int argc,char **argv){
    double ar = 1.0, br = 2.0, cr = 3.0;
    double ae = 2.0, be = -1.0, ce = 5.0;
    int N = 300;
    double w_sigma = 1.0;
    double inv_sigma = 1.0/ w_sigma;
    cv::RNG rng;

    vector<double> x_data, y_data;
    for(int i = 0;i < N;i++){
        double x = i/100.0;
        x_data.push_back(x);
        y_data.push_back(exp(ar * x* x + br * x+ cr) +
rng.gaussian(w_sigma*w_sigma));
    }
}


```

- 模块化的ceres处理方式
- [Why? — Ceres Solver \(ceres-solver.org\)](http://ceres-solver.org)

```

    double abc[3] = {ae, be, ce};

    ceres::Problem problem;
    for(int i = 0;i<N;i++){
        problem.AddResidualBlock(
            new ceres::AutoDiffCostFunction<CURVE_FITTING_COST, 1, 3>(
                new CURVE_FITTING_COST(x_data[i],y_data[i])
            ),
            nullptr,
            abc
        );
    }

    ceres::Solver::Options options;
    options.linear_solver_type = ceres::DENSE_NORMAL_CHOLESKY;
    options.minimizer_progress_to_stdout = true;

    ceres::Solver::Summary summary;
    chrono::steady_clock::time_point t1 = chrono::steady_clock::now();
    ceres::Solve(options, &problem, &summary);
    chrono::steady_clock::time_point t2 = chrono::steady_clock::now();
    chrono::duration<double> time_used =
    chrono::duration_cast<chrono::duration<double>>(t2-t1);
    cout << "solve time cost = " << time_used.count() <<" seconds."
    <<endl;
}
}

```

- 查看结果报告的基本操作

```

    cout << summary.BriefReport()<<endl;
    cout << "estimated a,b,c = ";
    for(auto a:abc) cout<< a<< " ";
    cout<<endl;

    return 0;
}
}

```

- 数据呈现:

iter	cost	cost_change	gradient	step	tr_ratio
tr_radius	ls_iter	iter_time	total_time		
0	8.233989e+03	0.00e+00	1.90e+04	0.00e+00	0.00e+00
1.00e+04	0	8.82e-06	2.91e-05		
1	9.001700e+02	7.33e+03	2.65e+03	8.65e-01	8.91e-01

```

1.91e+04      1   8.01e-05   1.68e-04
    2   6.760527e+01   8.33e+02   3.69e+02   6.79e-01   9.25e-01
4.95e+04      1   8.11e-06   2.31e-04
    3   1.792865e+00   6.58e+01   4.16e+01   3.67e-01   9.73e-01
1.48e+05      1   5.01e-06   2.46e-04
    4   3.143448e-03   1.79e+00   1.60e+00   8.62e-02   9.98e-01
4.45e+05      1   4.05e-06   2.58e-04
    5   1.214630e-08   3.14e-03   3.13e-03   3.93e-03   1.00e+00
1.34e+06      1   2.86e-06   2.69e-04
    6   2.601730e-19   1.21e-08   1.45e-08   7.76e-06   1.00e+00
4.01e+06      1   5.96e-06   3.06e-04
solve time cost = 0.000333696 seconds.
Ceres Solver Report: Iterations: 7, Initial cost: 8.233989e+03, Final
cost: 2.601730e-19, Termination: CONVERGENCE
estimated a,b,c = 2 -1 3

```

高斯牛顿法

利用两次的一阶梯度来代替二阶梯度的求解，有效降低求梯度的计算量

- 高斯牛顿法代码实现
- cmakelist

```

cmake_minimum_required(VERSION 3.2)

project(gaussNewton)

find_package(OpenCV REQUIRED)

set(CMAKE_CXX_FLAGS "-std=c++14")

set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)

aux_source_directory(src SRC_LIST)

include_directories(include)

include_directories("/usr/include/eigen3")

add_executable(gaussNewton ${SRC_LIST})

target_link_libraries(gaussNewton ${OpenCV_LIBS})

```

- cpp file as follow
- 构造方法与ceres一样

```
#include <iostream>
#include <opencv2/opencv.hpp>
#include <eigen3/Eigen/Core>
#include <eigen3/Eigen/Dense>

using namespace std;
using namespace Eigen;

int main(int argc, char **argv){
    double ar = 1.0, br = 2.0, cr = 1.0;
    double ae = 2.0, be = -1.0, ce = 5.0;

    int N = 100;
    double w_sigma = 1.0;
    double inv_sigma = 1.0 / w_sigma;
    cv::RNG rng;

    vector<double> x_data, y_data;
    for(int i = 0; i < N; i++){
        double x = i / 100;
        x_data.push_back(x);
        y_data.push_back(ar * x * x + br * x + cr +
rng.gaussian(w_sigma * w_sigma));
    }
}
```

- 利用两重循环解决迭代问题
-

```
int iterations = 100;
double cost = 0, lastCost = 0;

chrono::steady_clock::time_point t1 = chrono::steady_clock::now();
for(int iter = 0; iter < iterations; iter++){

    Matrix3d H = Matrix3d::Zero();
    Vector3d b = Vector3d::Zero();

    cost = 0;

    for(int i = 0; i < N; i++){
        double xi = x_data[i], yi = y_data[i];
    }
}
```

```

        double error = yi - exp(ae * xi * xi + be * xi + ce);
        Vector3d J;
        J[0] = -xi * xi * exp(ae * xi * xi + be * xi + ce);
        J[1] = -xi * exp(ae * xi * xi + be * xi + ce);
        J[2] = - exp(ae * xi * xi + be * xi + ce);

        H += inv_sigma * inv_sigma * J *J.transpose();
        b += -inv_sigma * inv_sigma * error * J;

        cost += error * error;
    }

    Vector3d dx = H.ldlt().solve(b);
    if(isnan(dx[0])){
        cout<<" result is nan!"<<endl;
        break;
    }

    if(iter > 0 && cost >= lastCost){
        cout<<" cost: "<<cost<<" >= lastCost: "<<lastCost<<",
break."<<endl;
        break;
    }

    ae += dx[0];
    be += dx[1];
    ce += dx[2];

    lastCost = cost;

    cout << "total cost: "<<cost<<, \t\t update: "<<dx.transpose()
<<
            "\t\testimated params:<< ae <<, "<<be<<, "<<ce<<endl;
    }
}

chrono::steady_clock::time_point t2 = chrono::steady_clock::now();
chrono::duration<double> time_used =
chrono::duration_cast<chrono::duration<double>>(t2 - t1);

cout<<"solve time cost: "<<time_used.count()<<"seconds."<<endl;
cout<<"estimated abc = " << ae <<, "<<be<<, "<<ce<<endl;

return 0;

```

```
}
```

- 数据显示

```
total cost: 2.17109e+06,           update:      0          0 -0.992786
estimated params:2,-1,4.00721
total cost: 290872,           update:      0          0 -0.980531
estimated params:2,-1,3.02668
total cost: 38356.9,           update:      0          0 -0.948098
estimated params:2,-1,2.07859
total cost: 4897.31,           update:      0          0 -0.866052
estimated params:2,-1,1.21253
total cost: 630.25,           update:      0          0 -0.681537
estimated params:2,-1,0.530996
total cost: 144.919,           update:      0          0 -0.370426
estimated params:2,-1,0.16057
total cost: 106.306,           update:      0          0 -0.0881557
estimated params:2,-1,0.0724144
total cost: 105.237,           update:      0          0
-0.00412181 estimated params:2,-1,0.0682925
total cost: 105.235,           update:      0          0
-8.51803e-06estimated params:2,-1,0.068284
total cost: 105.235,           update:      0          0
-3.62787e-11estimated params:2,-1,0.068284
cost: 105.235 >= lastCost: 105.235, break.
solve time cost: 0.00481325seconds.
estimated abc = 2,-1,0.068284
```

阻尼牛顿法

$$(H + \lambda I)\Delta x_k = g$$

阻尼牛顿法是高斯牛顿法和一阶梯度法之间的权衡，当I占主导则说明二次近似不够好，而H占主导说明二次近似比较好