**N1移动平台调试开发指南**

目录

[一、N1功能介绍 4](#_Toc525135699)

[1.1 N1系统框架介绍 4](#_Toc525135700)

[1.2 N1导航软件功能介绍 4](#_Toc525135701)

[二、N1导航系统主要参数详解 5](#_Toc525135702)

[2.1 N1底盘基础参数介绍及调试方法 5](#_Toc525135704)

[测试一：前进1米 5](#_Toc525135705)

[测试二：原地转动360度 6](#_Toc525135706)

[底盘参数调试方法 6](#_Toc525135707)

[底盘关键数据观察 7](#_Toc525135708)

[2.2 N1底盘上下两雷达安装及参数调试 8](#_Toc525135709)

[N1 上下两雷达安装情况 8](#_Toc525135710)

[雷达具体参数介绍 9](#_Toc525135711)

[雷达参数调试的方法 10](#_Toc525135712)

[雷达关键数据观察 12](#_Toc525135713)

[2.3 N1超声波的安装及参数介绍 12](#_Toc525135714)

[超声波的位置安装（目前默认位置） 12](#_Toc525135715)

[超声波参数介绍 13](#_Toc525135716)

[超声波参数调试方法 14](#_Toc525135717)

[超声波关键数据观察 15](#_Toc525135718)

[2.4 建图与导航参数介绍及调试 15](#_Toc525135719)

[建图与导航使用 18](#_Toc525135720)

[建图与导航参数调节的情况 19](#_Toc525135721)

[2.5 建图与导航主要数据观察 20](#_Toc525135722)

[2.6底盘自动回充功能 21](#_Toc525135723)

[环境要求： 21](#_Toc525135724)

[方法一：导航过程中，导航模块给底盘下发指令让底盘开始回充 22](#_Toc525135725)

[方法二：用Web APP—EGO 建图和导航，在WebAPP上启动回充充电. 22](#_Toc525135726)

[2.7 Sick 雷达建图与导航 24](#_Toc525135727)

[Sick 雷达建图导航使用 24](#_Toc525135728)

[2.8 深度摄像头导航避障 25](#_Toc525135729)

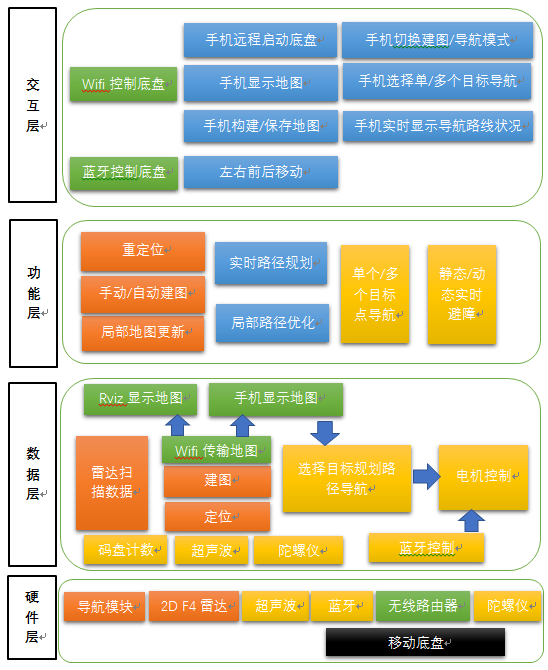
[深度摄像头参数及校准 25](#_Toc525135730)

[深度摄像头导航避障 26](#_Toc525135731)

[修订历史 26](#_Toc525135732)

# 一、N1功能介绍

## N1系统框架介绍



## N1导航软件功能介绍

|  |  |  |
| --- | --- | --- |
| 功能 | 支持功能 | 功能说明 |
| 建图功能 | Gmapping算法建图 | 适合一般室内环境建图（150\*150平方米以内） |
| 导航功能 | 双雷达导航 | 增加安全性，有效避开矮的障碍物，防止压脚 |
| 超声波导航避障 | 增加安全性，有效避开玻璃等透明障碍物 |
| 深度摄像头导航避障 | 增加安全性，达到3维避障效果 |
| 融合陀螺仪 | 提高导航精度，减小累计误差 |
| 单点导航，多点巡逻导航 | 多种导航方式，提高场景适应性 |
| 客户端APP | 通过（网络）wifi启动底盘建图，并显示地图，保存地图，然后导航 | 直接在手机上完成建图，导航使用，提高适用性，降低使用门槛 |
| 通过APP查看底盘各种传感器状态和日志信息 |
| 自动回充功能 | 增加智能性，有效续航，方便长时间使用 |

# 二、N1导航系统主要参数详解



## 2.1 N1底盘基础参数介绍及调试方法

底盘基础参数主要是底盘的轮子直径，两轮子的间距，编码器值以及底盘移动速度控制，具体如下：

|  |  |  |
| --- | --- | --- |
| 参数文件路径：dashgo\_ws/src/dashgo/dashgo\_driver/config/my\_dashgo\_params\_imu.yaml | | |
| 参数名 | 目前值 | 说明 |
| wheel\_diameter | 0.148 | 轮子直径，固定 |
| wheel\_track | 0.445 | 两个轮子的间距，直接测量得到大概值，然后再通过转360度试验进行细微调整，出厂时会把参数调好，直接使用即可。 |
| encoder\_resolution | 610 | 轮子转动一圈编码器输出的脉冲数，固定 |
| gear\_reduction | 0.98 | 校准系数，主要用于校准走1m直线 |

通过测试底盘走1m 直线，360度转。

### 测试一：前进1米

远程进入导航模块，启动底盘驱动(带陀螺仪)，

$ ssh eaibot@192.168.31.200

$ roslaunch dashgo\_driver driver\_imu.launch

然后远程进入导航模块另一个终端，启动移动脚本，

$ ssh eaibot@192.168.31.200

$ rosrun dashgo\_tools check\_linear\_imu.py

测试完后，ctl+c结束两个终端的程序。

### 测试二：原地转动360度

远程进入导航模块，启动底盘驱动(带陀螺仪)，

$ ssh eaibot@192.168.31.200

$ roslaunch dashgo\_driver driver\_imu.launch

然后远程进入导航模块另一个终端，启动转动脚本。

$ ssh eaibot@192.168.31.200

$ rosrun dashgo\_tools check\_angular\_imu.py

### 底盘参数调试方法

#### 情况一:走1m 直线和360度旋转参数调试情况

优先校准走1m直线，仅需修改my\_dashgo\_params.yaml文件中的gear\_reduction参数，其他参数基本给定，超过1m时，就改小，否则就改大，误差控制在1%左右。

在已校准1m 直线的前提下，校准360度旋转，仅需细微修改两个轮子间距wheel\_track，转超360度就改小，否则就改大，误差控制在1%左右。

如果底盘走成了斜直线，一般都是底盘摆放时，两轮子不在同一水平线引起的。Eai底盘在出厂时，都会测试确认底盘能正常走直线。

如果底盘走S型（无法走直线），先确认底盘是否电量充足，若电量不足，无法拉动电机正常转动，就会行走异常，若充足但行走异常请找eai售后

#### 情况二：提高/限制底盘移动速度

启动driver.launch或driver\_imu.launch来驱动底盘行走时，默认都是有平缓控制速度的功能，所以主要是修改yocs\_velocity\_smoother.yaml配置文件中的最大线速度speed\_lim\_v和最大角速度speed\_lim\_w来控制底盘行走的。

如果是在启动导航navigation 时，此时底盘行走速度不单受平缓速度yocs\_velocity\_smoother.yaml的限制，还会受到局部路径规划teb\_local\_planner\_params.yaml中的最大线速度限制，最终会取两者中最小的线速度。这点会在导航章节中详述。

如果是只用手机蓝牙来控制底盘时，速度是不受平缓控制的，所以会有急停（点头）和急速前冲（抬头）的现象。

### 底盘关键数据观察

以使用三—键盘控制底盘行走为例，主要观察底盘的里程计信息，线速度，角速度信息。具体如下：

#### odom—不带陀螺仪的里程计信息

.在启动driver.launch的情况下，在导航模块的另一个终端中输入指令，

rostopic echo /odom

#### odom\_combined，imu，imu\_angle—陀螺仪和带陀螺仪的里程计信息

在启动driver\_imu.launch的情况下，会把/odom的信息与陀螺仪/imu的信息融合后得到新的里程计信息，并发出来给建图导航使用，在导航模块的另一个终端中分别输入指令。

rostopic echo /robot\_pose\_ekf/odom\_combined #查看带陀螺仪里程计信息

rostopic echo /imu

rostopic echo /imu\_angle #查看陀螺仪角度变化信息

#### /smoother\_cmd\_vel—经过平缓处理的底盘速度

在启动driver.launch或driver\_imu.launch驱动底盘的情况下，底盘的最原始的速度信息是在/cmd\_vel中，经过平缓处理后，发布到新的主题/smoother\_cmd\_vel，建图导航等默认都使用经过平缓处理后的速度，在导航模块的另一个终端中分别输入指令。

rostopic echo /cmd\_vel #底盘原始的速度信息

rostopic echo /smoother\_cmd\_vel #经过平缓处理后的速度，默认底盘驱动，建图，导航等都是用这里的线速度和角速度，然后在dashgo\_driver.py 中把线速度和角速度转换成点击的pwd值发给底盘从而控制底盘行走

## 2.2 N1底盘上下两雷达安装及参数调试

### N1 上下两雷达安装情况

N1 上雷达为正面安装，如下图所示，数据线口为正后方，最大扫描角度为360度，受外壳开模影响，上雷达至少要180~270度的扫描角度，否则会影响扫图避障功能。雷达的具体参数，性能及单独使用请参照《雷达使用手册》。



N1 下面雷达为倒装雷达，如下图所示，数据线口为正后方，目前仅适用180度扫描角度，该雷达仅做导航避障使用。雷达的具体参数，性能及单独使用请参照《雷达使用手册》。



### 雷达具体参数介绍

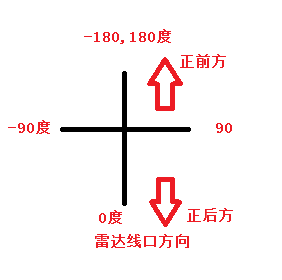
|  |
| --- |
| 参数文件路径：dashgo\_ws/src/dashgo/ydlidar-1.3.1/launch/ydlidar1\_up.launch  1.3.1为雷达驱动版本号，有可能变化, ydlidar1\_up.launch为右前方上雷达1启动文件，ydlidar2\_backup.launch为左后方上雷达2，ydlidar3\_down.launch为下雷达启动文件。 |
| <node name="ydlidar\_node" pkg="ydlidar" type="ydlidar\_node" output="screen">  雷达启动的节点名为ydlidar\_node。 |
| <param name="port" type="string" value="/dev/port2"/>  雷达与导航模块连接的串口，为port2。 |
| <param name="baudrate" type="int" value="230400"/>  G4 雷达串口波特率，如果是F4雷达则为115200，X4雷达为128000。 |
| <param name="angle\_min" type="double" value="-135" />  <param name="angle\_max" type="double" value="135" />  雷达扫描角度范围为-180~180度，雷达角度范围设置具体参照下面说明。 |
| <param name="range\_min" type="double" value="0.08" />  <param name="range\_max" type="double" value="16.0" />  雷达扫描距离范围为0.08~16 m。 |
| <param name="ignore\_array" type="string" value=" " />  雷达剔除的扫描范围，即不取该范围的数据，它与上面扫描角度参数结合，得到最终雷达有效的扫描角度范围。 |
| <node pkg="tf" type="static\_transform\_publisher" name="base\_link\_to\_laser4"  args="0.174 -0.176 0.28 -0.785 0.0 0.0 /base\_footprint /laser\_frame 40" />  这是右前方上雷达（正面摆放的雷达）与N1底盘的tf转换参数。  如果是下雷达，tf转换参数如下：  <node pkg="tf" type="static\_transform\_publisher" name="base\_link\_to\_laser4\_2"  args="0.17 0.0 0.05 -3.04 3.14 0.0 /base\_footprint /laser\_frame\_2 40" /> |

|  |
| --- |
| 参数文件路径：dashgo\_ws/src/dashgo/dashgo\_tools/conf/box\_filter.yaml  其中box\_filter.yaml表示右前方上雷达安全范围，box\_filter\_2.yaml为下雷达安全范围。 |
| max\_x: 0.45安全范围在x 轴上离底盘重心最大距离  max\_y: 0.32安全范围在y轴左边离底盘重心距离  max\_z: 0.5暂时无用  min\_x: 0.1安全范围在x 轴上离底盘重心最小距离  min\_y: -0.32安全范围在y轴右边离底盘重心距离  min\_z: 0.05暂时无用  参数具体意义见下面情况三详解 |

### 雷达参数调试的方法

#### 情况一：设置雷达的扫描角度

设置雷达的扫描角度并剔除在扫描范围特定角度的数据（如只取雷达前方270度数据或者剔除扫描范围内的柱子等物体），雷达的数据获取符合右手定则（与雷达的转动方向没直接关系），具体如下图所示：



如果雷达只想扫描正前方270度，则需要把ydlidar1\_up.launch的参数设置如下（以上雷达为例）：

<param name="angle\_min" type="double" value="-180" />

<param name="angle\_max" type="double" value="180" />

<param name="ignore\_array" type="string" value="-45,45" />

剔除雷达-45 到 45 度的数据

注意：雷达的扫描角度不能小于180度，否则会影响建图，导航避障等功能。

#### 情况二：雷达坐标系与底盘坐标系的tf转换关系设置

该参数主要是在整套移动系统在建图导航前，进行雷达校准用到，单独雷达不需要用到此参数。

以上雷达为例，雷达正装，则ydlidar1\_up.launch参数设置如下：（一般出厂时雷达参数都会设置好，可直接使用，但若移动，拆装后需要自己细微调整）

<node pkg="tf" type="static\_transform\_publisher" name="base\_link\_to\_laser4"

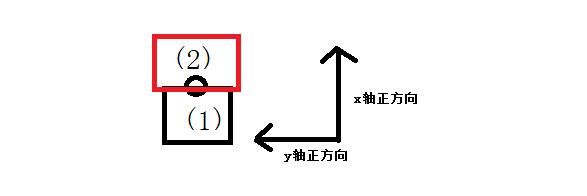
args="0.174 -0.176 0.28 -0.785 0.0 0.0 /base\_footprint /laser\_frame 40" />

* args第一个参数0.174表示雷达中心距离底盘重心的x 轴距离；
* args第二个参数-0.176 表示雷达中心距离底盘重心的y轴距离；
* args第二个参数0.28 表示雷达中心距离底盘重心的z轴距离，该参数为虚拟的，不能改，因为会影响到导航的costmap，（因为G4雷达为2维雷达，z轴参数对雷达数据没影响，所以可以使用虚拟）；
* args第四个参数表示将雷达绕z 轴左右偏转程度，为yaw 偏航角；
* args第五个参数表示将雷达绕y轴前后翻滚程度，为pitch 俯仰角；
* args第六个参数表示将雷达绕x 轴左右侧滚，为roll 侧滚角，该参数一般为0.0，

目前只能设为0.0，-3.14 和3.14。

#### 情况三：雷达滤波安全范围设置（仅在导航时使用）

如下图所示，雷达滤波安全范围是指，在雷达前方，画一个安全区域，一旦雷达突然发现前方有障碍物出现在安全区域内（例如底盘导航时，突然伸脚到底盘前面很近的地方），此时底盘优先停下然后再重新规划路径绕开，它认为离突然出现的障碍物太近，再往前就会撞到障碍物，这样可以有效防止底盘减速刹车不及时撞到障碍物的问题。



如图所示，红色部分为安全范围，它为矩形，根据box\_filter.yaml参数，以底盘重心为原点，正常安全范围（红色部分）x轴长度在15cm 左右，y轴宽度比底盘宽2cm（左右两侧各宽1cm），注意：该安全范围不能过大，否则会影响导航效果（例如通过狭窄的地方）。

### 雷达关键数据观察

主要是观察/scan 雷达节点是否有数据。

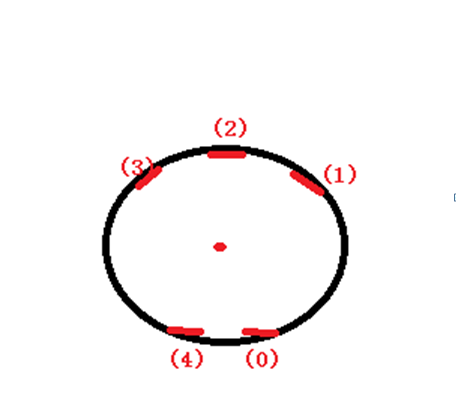
rostopic echo /scan

## 2.3 N1超声波的安装及参数介绍

目前N1系统最多只支持6个超声波，现在N1安装了5个超声波的，安装在stm32 控制板上（电机控制板），stm32 会实时把超声波数据传给导航模块，然后和导航避障算法进行融合避障。

### 超声波的位置安装（目前默认位置）

N1 超声波默认的安装情况如下：



0号在后方右侧，离中心坐标为（-0.1945, -0.11），偏角为-3.14 弧度；

1号在前方右侧，离中心坐标为（0.1868,-0.23），偏角为-0.872弧度；

2号在正前方，离中心坐标为（0.23,0.0），偏角为0 弧度；

3号在前方左侧，离中心坐标为（0.1868,0.23），偏角为0.872弧度。

4号在后方左侧，离中心坐标为（-0.1945, 0.11），偏角为3.14 弧度；

### 超声波参数介绍

|  |
| --- |
| 参数文件路径：dashgo\_ws/src/dashgo/dashgo\_driver/config/my\_dashgo\_params\_imu.yaml  主要是超声波功能开关，坐标位置及偏移角初始化。 |
| useSonar: True  超声波功能开关，True表示打开，False 表示关闭。 |
| sonar0\_offset\_yaw: -3.14 5个超声波的位置和偏移角初始化  sonar0\_offset\_x: -0.1945  sonar0\_offset\_y: -0.11  sonar1\_offset\_yaw: -0.872  sonar1\_offset\_x: 0.1868  sonar1\_offset\_y: -0.23  sonar2\_offset\_yaw: 0.0  sonar2\_offset\_x: 0.23  sonar2\_offset\_y: 0.0  sonar3\_offset\_yaw: 0.872  sonar3\_offset\_x: 0.1868  sonar3\_offset\_y: 0.23  sonar4\_offset\_yaw: 3.14  sonar4\_offset\_x: -0.1945  sonar4\_offset\_y: 0.11 |

|  |
| --- |
| 参数文件路径：dashgo\_ws/src/dashgo/dashgo\_driver/launch/driver\_imu.launch |
| <node pkg="tf" type="static\_transform\_publisher" name="base\_link\_to\_sonar0"  args="-0.1945 -0.11 0.15 -3.14 0.0 0.0 /base\_footprint /sonar0 40" />  <node pkg="tf" type="static\_transform\_publisher" name="base\_link\_to\_sonar1"  args="0.1868 -0.23 0.15 -0.872 0.0 0.0 /base\_footprint /sonar1 40" />  <node pkg="tf" type="static\_transform\_publisher" name="base\_link\_to\_sonar2"  args="0.23 0.0 0.15 0.0 0.0 0.0 /base\_footprint /sonar2 40" />  <node pkg="tf" type="static\_transform\_publisher" name="base\_link\_to\_sonar3"  args="0.1868 0.23 0.15 0.872 0.0 0.0 /base\_footprint /sonar3 40" />  <node pkg="tf" type="static\_transform\_publisher" name="base\_link\_to\_sonar4"  args="-0.1945 0.11 0.15 3.14 0.0 0.0 /base\_footprint /sonar4 40" />  这分别是超声波与底盘的坐标tf转换关系，同时会把相应位置显示到rviz上，如果rviz上看到超声波位置不对，请查看此参数是否正确。 |

### 超声波参数调试方法

步骤一:要使用超声波避障功能，必须先打开超声波功能开关。

useSonar: True

超声波功能开关，True表示打开，False 表示关闭。

步骤2:运行导航launch,观察超声波数据变化，验证每一个超声波工作正常具体如下：

在导航模块中运行导航launch（以不带陀螺仪单点导航为例）；

roslaunch dashgo\_nav navigation\_imu\_2.launch

在导航模块另一个终端中，分别监听每一个超声波主题，如下与0号超声波为例；

rostopic echo /sonar0

然后再0号超声波前面放一个障碍物并来回移动，观察0号超声波数据变化是否正常，

以此验证其他四个超声波是否都正常。

步骤3：在rviz中添加超声波的显示，并观察超声波看到障碍物时，是否会停止。

保持运行导航navigation\_imu\_2.launch程序，在电脑终端中运行rviz显示地图，add->by Topic 然后选择sonar0 的Range,点击ok 就会把超声波的锥形范围显示在rviz中，最后ctrl+s保存rviz配置，类似地把其他超声波加入到rviz中，然后再把障碍物放到超声波前面（障碍物最好是玻璃等透明物体），观察导航时是否避开它。

### 超声波关键数据观察

主要是观察/sonar0~4超声波节点是否有数据。

rostopic echo /sonar0 观察0号超声波数据，默认情况只取0.8m 以内的有效数据

rostopic echo /sonar1

rostopic echo /sonar2

rostopic echo /sonar3

rostopic echo /sonar4

## 2.4 建图与导航参数介绍及调试

|  |
| --- |
| 参数文件路径：dashgo\_ws/src/dashgo/dashgo\_nav/launch/include/imu/gmapping\_base.launch  Gmapping扫图算法参数。 |
| <param name="maxUrange" value="8.0"/>  <param name="maxRange" value="10.0"/>  雷达最远扫描距离设置，正常G4 能扫到16m，由于越远激光数据点越少且不稳定，因此只取10m 内的数据。 |
|  |
| 参数文件路径：dashgo\_ws/src/dashgo/dashgo\_nav/config/imu/ teb\_local\_planner\_params.yaml  Teb局部路径规划配置： |
| max\_vel\_x: 0.6  #机器人导航时最大线速度，与1.2.4章节中情况三控制底盘平缓行走的参数一起控制底盘导航，最终取两者最小的线速度。  max\_vel\_x\_backwards: 0.15 #机器人后退速度，不能改小  max\_vel\_theta: 0.30 #最大角速度  acc\_lim\_x: 0.30 #线加速度  acc\_lim\_theta: 0.30 #角加速度，不能过大，否则行走可能左右摆动  min\_turning\_radius: 0.0  footprint\_model: # types: "point", "circular", "two\_circles", "line", "polygon"  vertices: [[-0.283, -0.283], [-0.283, 0.283],[0.283, 0.283], [0.283, -0.283]]  #底盘模型  # GoalTolerance  xy\_goal\_tolerance: 0.2 #导航里目标点最大距离误差为20cm  yaw\_goal\_tolerance: 0.5 #最大角度误差为0.5 \*6=30度  free\_goal\_vel: False  # Obstacles  min\_obstacle\_dist: 0.32 #距离障碍物的最小距离  weight\_kinematics\_forward\_drive: 100 #机器人前进的权重，增大时，机器人后退几率，后退距离都会减小，但不能过大，具体要根据实际情况调试。 |
|  |
| 参数文件路径：dashgo\_ws/src/dashgo/dashgo\_nav/config/imu/ move\_base\_params.yaml  Move\_base算法参数： |
| planner\_frequency: 1.0 #路径规划频率  oscillation\_timeout: 5.0 #超时时间为5.0\*2=10s  oscillation\_distance: 0.2 #如果在10s（超时时间）内，机器人没有行走超过0.2m，则认为机器人在来回挪动（震荡），此时取消该次导航 |
|  |
| 参数文件路径：dashgo\_ws/src/dashgo/dashgo\_nav/config/imu/ costmap\_common\_params.yaml  代价地图costmap基础参数： |
| footprint: [[-0.283, -0.283], [-0.283, 0.283],[0.283, 0.283], [0.283, -0.283]] #底盘的模型  obstacle\_layer: #动态层costmap  enabled: true  max\_obstacle\_height: 1.2 #costmap的最大高度  min\_obstacle\_height: 0.0  obstacle\_range: 2.0 #2m 内有障碍物就加入costmap中  raytrace\_range: 5.0  inflation\_radius: 0.30 #障碍物膨胀系数  combination\_method: 1  #非常重要，这里表明costmap是由传感器雷达laser\_scan\_sensor，超声波  #sonar\_scan\_sensor数据组成，数据来源具体下面会介绍  observation\_sources: laser\_scan\_sensorsonar\_scan\_sensor  track\_unknown\_space: true #是否往未知区域规划路径  origin\_z: 0.0 #costmap高度从0m开始  z\_resolution: 0.1 #costmap立体分成，每一层为0.1m  z\_voxels: 10 #costmap立体一共分10 层数据  unknown\_threshold: 15  mark\_threshold: 0  publish\_voxel\_map: true  footprint\_clearing\_enabled: true #是否清楚底盘脚下的costmap  laser\_scan\_sensor: #表明是从/scan 主题中获取雷达数据构成costmap  data\_type: LaserScan #右上雷达数据类型  topic: /scan #右上雷达数据主题  marking: true  clearing: true  expected\_update\_rate: 0  min\_obstacle\_height: 0.21 #右上雷达数据在costmap中的高度范围  max\_obstacle\_height: 0.30  laser\_scan\_sensor\_2: #底下雷达数据  data\_type: LaserScan  topic: /scan\_2  marking: true  clearing: true  expected\_update\_rate: 0  min\_obstacle\_height: 0.01  max\_obstacle\_height: 0.1  laser\_scan\_sensor\_3: #左后雷达  data\_type: LaserScan  topic: /scan\_3  marking: true  clearing: true  expected\_update\_rate: 0  min\_obstacle\_height: 0.31  max\_obstacle\_height: 0.40  sonar\_scan\_sensor: #超声波点云数据  data\_type: PointCloud2  topic: /sonar\_cloudpoint  marking: true  clearing: true  min\_obstacle\_height: 0.11 #超声波点云数据在costmap中高度范围  max\_obstacle\_height: 0.2  inflation\_layer: #静态层costmap  enabled: true  cost\_scaling\_factor: 10.0 # exponential rate at which the obstacle cost drops off (default: 10)  inflation\_radius: 0.30 # 障碍物膨胀系数  static\_layer:  enabled: true  map\_topic: "/map"  sonar\_layer: #超声波数据  enabled: true  clear\_threshold: 0.2  mark\_threshold: 0.8  topics: ["/sonar0", "/sonar1", "/sonar2", "/sonar3", "/sonar4"]  clear\_on\_max\_reading: true |

### 建图与导航使用

步骤1：在导航模块中，启动建图launch

$ ssh [eaibot@192.168.31.200](mailto:eaibot@192.168.31.200) #远程进导航模块

$ roslaunch dashgo\_nav gmapping\_imu.launch

步骤2：在电脑ubuntu系统中，启动rviz工具（注意该命令是在电脑上运行，而不是导航模块中，之后启动rviz的操作都是在电脑上，导航模块中没有安装rviz工具）

$ export ROS\_MASTER\_URI=http://192.168.31.200:11311

$ roslaunch dashgo\_rviz view\_navigation.launch

步骤3：手机app wifi控制底盘行走（注意此时不能用蓝牙控制，会导致控制冲突）

步骤4：建完地图后，保持建图程序运行，进行如下操作保存好地图

$ ssh eaibot@192.168.31.200 　　#远程进入导航模块

$ roscd dashgo\_nav/maps 　　 #进入地图目录

$ rosrun map\_server map\_saver -f eai\_map\_imu

#保存地图，名为eai\_map\_imu,然后会在maps目录下生成eai\_map\_imu.yaml和eai\_map\_imu.pgm文件（即保存的地图为pgm格式），之后带陀螺仪导航时，默认会导入名为eai\_map\_imu的地图，

#### 地图保存好后，ctl+c关闭建图程序

步骤5：在导航模块中，启动单点导航的launch

$ ssh [eaibot@192.168.31.200](mailto:eaibot@192.168.31.200)

$ roslaunch dashgo\_nav navigation\_imu\_2.launch

步骤6：在电脑ubuntu系统中，启动rviz工具

$ export ROS\_MASTER\_URI=http://192.168.31.200:11311

$ roslaunch dashgo\_rviz view\_navigation.launch

步骤7：设置机器人起点位置，然后设置单个目标位置，开始导航

### 建图与导航参数调节的情况

#### 情况一：修改障碍物膨胀系数，防止规划的路径贴近障碍物（沿边规划）

主要修改costmap\_common\_params.yaml文件中的inflation\_radius参数，该文件中有两个这样的参数，必须同时该

inflation\_radius: 0.30 #障碍物膨胀系数

obstacle cost drops off (default: 10)

inflation\_radius: 0.30 # 障碍物膨胀系数

#### 情况二：限制机器人导航行走的速度

主要修改局部路径规划teb\_local\_planner\_params.yaml的参数，它与1.2.4章节情况三—平缓速度限制一起控制底盘，要想限制机器人行走速度，需要同时修改两个配置文件，最终会取两者中最小的线速度和角速度

# Robot

max\_vel\_x: 0.6 #机器人导航时最大线速度，与1.2.4章节中情况三控制底盘平缓行走的参数一起控制底盘导航，最终取两者最小的线速度。

max\_vel\_x\_backwards: 0.15 #机器人后退速度

max\_vel\_theta: 0.30 #最大角速度

acc\_lim\_x: 0.30 #线加速度

acc\_lim\_theta: 0.30 #角加速度，不能过大，否则行走可能左右摆动

min\_turning\_radius: 0.0

#### 情况三：限制机器人只能前进，不能后退

主要修改局部路径规划teb\_local\_planner\_params.yaml中的机器人前进的权重参数，减小机器人后退几率和后退距离

weight\_kinematics\_forward\_drive: 100

#机器人前进的权重，增大时，机器人后退几率，后退距离都会减小，但不能过大，否则导航起步时可能会停止不动，具体要根据实际情况调试。修改该参数还没能使底盘完全不会后退的情况（尤其是在转180度时有可能会稍微后退调整），后续会继续优化

## 2.5 建图与导航主要数据观察

在启动导航launch情况下（例如roslaunch dashgo\_nav navigation\_imu\_2.launch）,然后rostopic list 列出所有的主题，如下分析常用关键的主题信息作用：

eaibot@PathGoE1:~$ rostopic list

/Lencoder #左轮编码器值变化

/Lvel #左轮速度

/Rencoder #右轮编码器值

/Rvel #右轮速度

/amcl\_pose #amcl算法定位得到底盘所处的地图位置

/cmd\_vel #下发给机器人的线速度和角速度

/emergencybt\_status #急停开关状态主题，1——按下，0——未按下

/imu #陀螺仪信息

/imu\_angle #陀螺仪的角度变化

/initialpose #导航时，默认的起点位置和方向

/is\_passed #显示在雷达滤波安全范围内是否有障碍物，>1 表示有障碍物，底盘线速度设为0，否则不影响导航

/is\_passed\_2 #显示在第二个（下雷达）滤波安全范围内是否有障碍物，只用在双雷达导航

/joint\_states #导航时，添加目标点是否成功状态反馈

#全局规划的路径，需要在rviz上才能直观地看到

/move\_base/TebLocalPlannerROS/global\_plan

#局部规划的路径，需要在rviz上才能直观地看到

/move\_base/cancel #取消当前导航

/move\_base/current\_goal #当前导航要去的目标点坐标

/move\_base/goal #当前导航要去的目标点坐标

/move\_base/result #导航结果反馈

/move\_base/status #导航实时状态反馈

/move\_base\_simple/goal #获取在rviz上点击设置的目标点坐标及方向

/odom #里程计信息

/robot\_cmd\_vel

/robot\_pose #机器人导航过程中的实时坐标及位姿信息

/robot\_pose\_ekf/odom\_combined #融合陀螺仪后，新的里程计信息

/scan #右前方上雷达数据

/scan\_2 #底下雷达数据

/scan\_3 #左后方上雷达数据

/smoother\_cmd\_vel #经过平缓处理后，发给底盘的速度信息

/sonar0 #0~4号超声波的数据

/sonar1

/sonar2

/sonar3

/sonar4

/sonar\_cloudpoint #0~4号超声波的点云数据

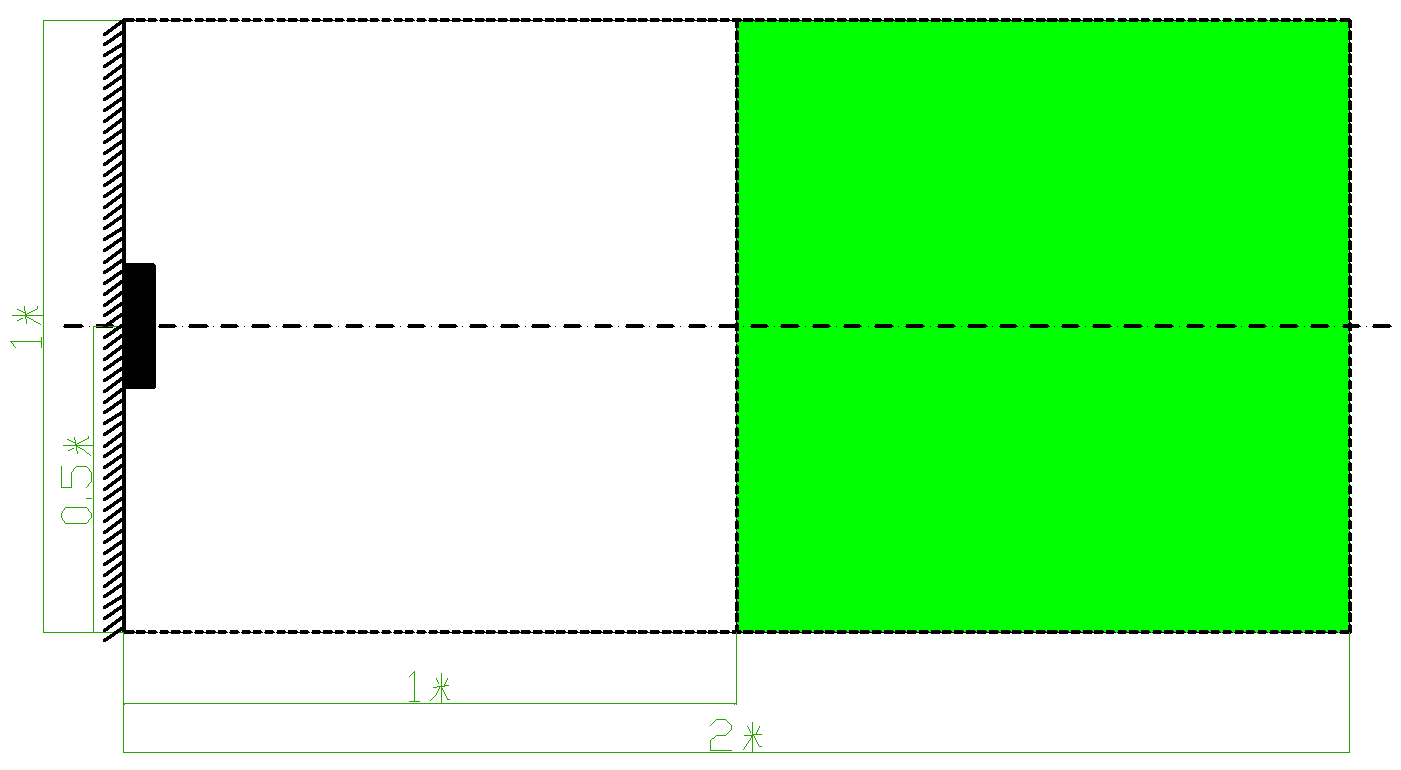
/voltage\_value #电量显示主题

## 2.6底盘自动回充功能

### 环境要求：

为确保自动回充功能正常使用，请您按要求设置充电环境：

1. 回充站必须与底盘在同一个平面内。
2. 回充站周围无类似镜面的反光物体。
3. 回充站需放置在左右分别有0.5米无障碍物空间的墙面。前方需有2米的无障碍物空间供底盘进行回充使用。其具体的要求如下图所示。



1. 注：绿色区域为最佳回充区域。

### 方法一：导航过程中，导航模块给底盘下发指令让底盘开始回充

例如，导航模块启动导航launch，

$ ssh [eaibot@192.168.31.200](mailto:eaibot@192.168.31.200)

$ roslaunch dashgo\_nav navigation\_imu\_2.launch

打开另一个终端，在ubuntu pc 中运行rviz观察地图

$ export ROS\_MASTER\_URI=http://192.168.31.200:11311

$ roslaunch dashgo\_rviz view\_navigation.launch

在rviz上设置好起点和适当的目标点（让底盘导航进入回充范围），等机器人导航到回充范围后，此时在导航模块的另外一个终端中，下发启动回充指令

$rostopic pub -1 /recharge\_handlestd\_msgs/Int16 1

此时会暂停导航，原地旋转进行红外对接，然后开始自动回充

若想停止，输入停止回充指令

$ rostopic pub -1 /recharge\_handlestd\_msgs/Int16 0

此时会退出回充状态，进入到导航状态，再一目标点后，就会继续导航

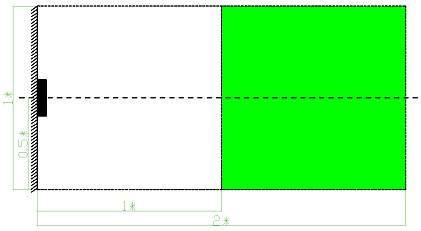
### 方法二：用Web APP—EGO 建图和导航，在WebAPP上启动回充充电.

用webapp建好地图后，切换到导航，如下图所示：



正常情况，切换到导航时，会自动设置好起点（如果发现起点位置不对，就需要手动设置初始位置的，具体操作看N1的使用手册），此时，先确定充电桩必须在所建的地图中。

先把底盘导航到充电桩附近的区域，就是导航到回充点的，回充点位置最好设置在充电桩正前方 1m左右的位置，如下图所示：黑色为充电桩位置的，绿色区域为最佳的回充点。



然后点击web页面的检验的“开始充电”，底盘的红外自动寻找充电桩，找到后开始充电对接，

注意：1.充电桩左右1米和前面2米范围内最好不要有其它东西的，保持空旷的，这样底盘红外感应效果比较好的，如果底盘是在绿色最佳的区域内启动回充，成功率达95%以上，如果不在绿色区域内，失败概率会增大（尤其是比较近充电桩时，失败率非常高，主要是没有足够距离来调整姿态）。

1. 2.正常回充成功会先滴一声表示回充到位，然后再滴两声表示正式开始充电，此时回充站绿灯闪烁。如果有其他情况，就表示不成功，需要放到合适位置，先发指令停止回充，若要继续回充，再要发启动回充指令。（例如电池比较满时，先滴一声表示回充到位，然后再滴两声表示正式开始充电，再滴一声表示电池满了，如果是滴6声，表示充电错误）

## 2.7 Sick 雷达建图与导航

Sick 雷达参数介绍

|  |
| --- |
| 参数文件路径:/opt/ros/kinetic/share/sick\_tim/launch/sick\_eai.launch |
| <param name="min\_ang" type="double" value="-1.57" />  <param name="max\_ang" type="double" value="1.57" />  雷达的扫描角度为-1.57~1.57弧度（即-90~90度）共180度，根据实际情况，最大只支持270度扫描角度  <param name="range\_max" type="double" value="25.0" />  sick雷达最大扫描距离  <param name="hostname" type="string" value="192.168.31.201" />  <param name="port" type="string" value="2112" />  <param name="timelimit" type="int" value="5" />  设置sick雷达通过网口把数据传给导航模块，并指定雷达ip为192.168.31.201，与导航模块ip必须在同一个网段,sick 雷达的ip地址需要在windows 下用sick官网软件设置，具体请参考sick 官网资料  <node pkg="tf" type="static\_transform\_publisher" name="base\_link\_to\_laser\_sick"  args="0.2245 0.0 0.28 0.08 0.0 0.0 /base\_footprint /sick\_laser 40" />  设置sick 雷达与底盘坐标系的tf转换关系，校准方法与eai雷达一样 |
|  |
| 参数文件路径：dashgo\_ws/src/dashgo/dashgo\_nav/launch/include/sick/imu/gmapping\_base.launch  Sick 雷达的建图与导航参数 |
| <param name="maxUrange" value="20.0"/>  <param name="maxRange" value="22.0"/>  Sick 雷达gmapping建图的扫描距离，不能都设置成25.0 |

### Sick 雷达建图导航使用

注意：1.sick 雷达使用前，必须进行tf坐标系校准（方法与F4雷达校准一样）

2.默认sick 雷达使用带陀螺仪建图与导航，根据情况可修改是否带陀螺仪

进入导航模块中，运行sick雷达建图launch，该launch 默认启动带陀螺仪建图

$ ssh eaibot@192.168.31.200

$roslaunch dashgo\_nav sick\_gmapping\_imu.launch

打开另一个终端，在ubuntu pc 中运行rviz观察地图

$ export ROS\_MASTER\_URI=http://192.168.31.200:11311

$ roslaunch dashgo\_rviz view\_navigation.launch

手机控制底盘行走，建好地图后，保持以上程序正常运行，打开另外一个终端，远程进导航模块，把地图保存在导航模块的dashgo\_nav/maps目录中，

$ ssh eaibot@192.168.31.200

$ roscd dashgo\_nav/maps/

$ rosrun map\_server map\_saver -f eai\_map\_imu

地图保存好后，ctrl+c退出建图程序，在导航模块中启动sick导航launch

$roslaunch dashgo\_nav sick\_navigation\_imu.launch

打开另一个终端，在ubuntu pc 中运行rviz观察地图

$ export ROS\_MASTER\_URI=http://192.168.31.200:11311

$ roslaunch dashgo\_rviz view\_navigation.launch

设置好起点位置，然后设置目标点开始导航.

## 2.8 深度摄像头导航避障

### 深度摄像头参数及校准

目前N1 支持奥比中光深度摄像头导航避障，具体参数调试如下:

|  |
| --- |
| 参数文件路径: ~/dashgo\_ws/src/dashgo/dashgo\_nav/launch/astra.launch |
| <node pkg="tf" type="static\_transform\_publisher" name="base\_link\_to\_camera"  args="0.25 0.0 0.20 0.08 0.0 0.0 /base\_footprint /camera\_link 40" />  摄像头与底盘的tf转换关系 |

在导航模块中，启动单独摄像头的launch：

roscd dashgo\_nav/launch

roslaunch astra.launch

然后在rviz中显示彩色点云，添加类型为PointClound2的topic，参数topic设置为/camera/depth\_registered/points，color transformer设置为rgb8，根据rviz中现实的图像可判断上述tf是否修改准确。

### 深度摄像头导航避障

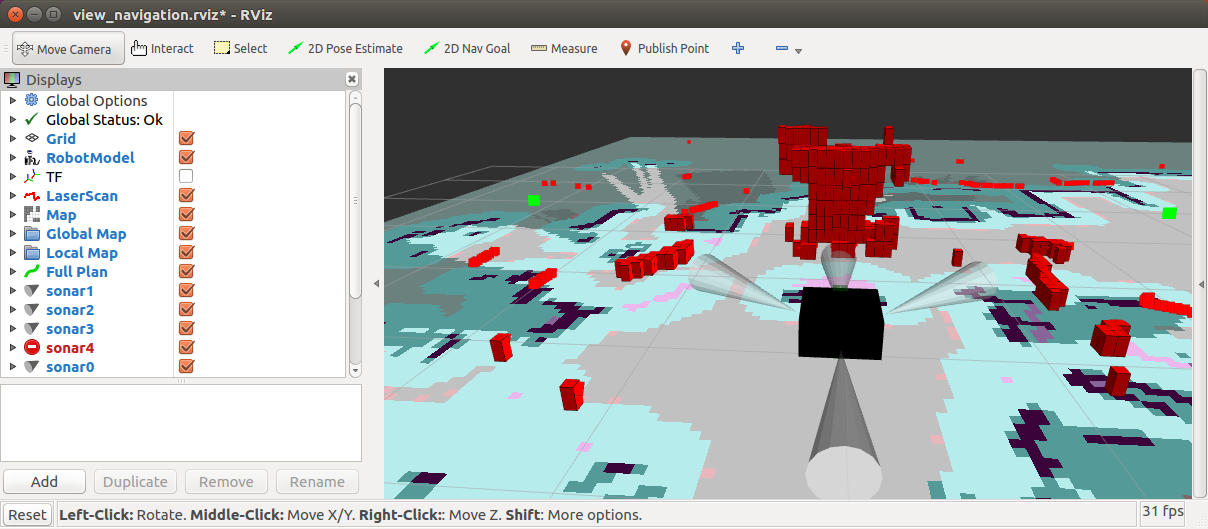
在导航模块中，启动带有摄像头导航的launch：

roslaunch dashgo\_nav navigation\_astra\_imu\_2.launch

在电脑ubuntu 中运行rviz，并添加PointCloub2 选择为/camera/depth\_registered/points

$ export ROS\_MASTER\_URI=http://192.168.31.200:11311

$ roslaunch dashgo\_rviz view\_navigation.launch



如图所示，摄像头会把看到的障碍物设置成多层的costmap，并让机器人导航时，避开这些障碍物。

# 修订历史

|  |  |
| --- | --- |
| 日期 | 内容 |
| 2018-07-16 | V1.0 |
| 2018-09-19 | V1.1 |