

## 目录

一、环境配置 .....	3
1 硬件部分: .....	3
1.1 移动平台 .....	3
1.2 IMU .....	3
1.3 激光雷达 .....	3
2 软件部分: .....	3
2.1 ROS 中需要安装的包 .....	4
2.2 EAI 移动平台相关的包 .....	4
2.3 IMU 相关 .....	4
2.4 激光雷达相关 .....	4
二、文件结构 .....	4
1. EAI 移动平台相关的包 .....	4
1.1dashgo_description.....	4
1.2dashgo_driver .....	5
1.3dashgo_tools.....	5
1.4dashgo_nav .....	5
1.5 dashgo_rviz .....	6
1.6navigation_task.....	6
2 IMU 相关 .....	6
2.1jy901b_imu .....	6
3 激光雷达相关 .....	7
3.1rplidar_ros.....	7
三、开发说明 .....	7
1 相关包的主要参数配置 .....	7
1.1 dashgo_driver .....	7
1.2 jy901b_imu .....	7
1.3 rplidar_ros.....	7
1.4 dashgo_nav .....	8
1.5navigation_task.....	13
2 导航节点 .....	13

2.1 程序流程 .....	13
2.2 目标点导航 .....	19
2.3 读取位置信息 .....	21
2.4 发布的消息 .....	22
2.5 订阅的消息 .....	22
2.6 消息回调函数 .....	22
2.7 service 服务.....	24
2.8 参数设置 .....	24
2.9 相关函数 .....	25
四、操作说明 .....	36
1 硬件校准 .....	36
1.1 移动平台校准 .....	36
1.2 IMU 校准: .....	36
1.3 激光雷达校准 .....	37
2 建图与导航 .....	37
2.1 dashgo 启动前检查.....	37
2.2 端口连接问题 .....	38
2.3 dashgo 控制.....	39
2.4 建立环境地图 .....	40
2.5 导航 .....	42

# 一、环境配置

## 1 硬件部分：

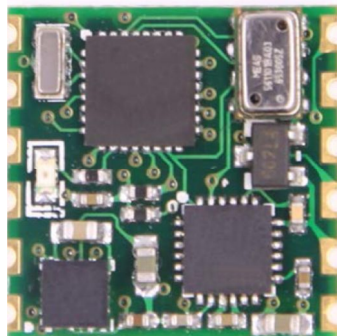
### 1.1 移动平台

EAI Dashgo D1：提供里程数据，接收速度命令。



### 1.2 IMU

JY-901：提供姿态数据



### 1.3 激光雷达

RPLIDAR A2：提供激光扫描数据



## 2 软件部分：

在安装完成 ubuntu 和 ros 之后，需要的软件包包括两个部分：

## 2.1 ROS 中需要安装的包

(通过 ubuntu 下的软件包安装命令安装)

ros-indigo-teb-local-planner : 局部规划器  
ros-indigo-teleop-twist-keyboard : 通过键盘发送速度命令  
ros-indigo-yocs-velocity-smoother : 速度平滑  
ros-indigo-gmapping : 建图  
ros-indigo-robot-pose-publisher : 机器人相关状态发布  
ros-indigo-move-base : 定位导航  
ros-indigo-navigation : 定位导航  
ros-indigo-rosbridge-server : rosbridge 的 WebSocket 接口  
ros-indigo-serial : 串口通信

## 2.2 EAI 移动平台相关的包

(将这些包拷贝到工作空间下编译)

- (1) dashgo\_description
- (2) dashgo\_driver
- (3) dashgo\_tools:
- (4) dashgo\_nav
- (5) dashgo\_rviz:
- (6) navigation\_task:

## 2.3 IMU 相关

(将这些包拷贝到工作空间下编译)

jy901b\_imu

## 2.4 激光雷达相关

(将这些包拷贝到工作空间下编译)

rplidar\_ros:

# 二、文件结构

## 1. EAI 移动平台相关的包

### 1.1 dashgo\_description

用于加载 EAI 模型。

启动命令: `roslaunch dashgo_description dashgo_description.launch`

主要文件 (或文件夹):

urdf/dashgobase:

- base.urdf.xacro
- dashgobase.xacro
- materials.urdf.xacro
- torso.urdf.xacro

EAI 移动平台相关的模型文件

launch:

dashgo\_description.launch:包含用于加载模型的启动文件

## 1.2dashgo\_driver

用于加载 EAI 平台驱动，设置了速度平滑。

启动命令：roslaunch dashgo\_driver demo.launch

主要文件（或文件夹）：

nodes:

dashgo\_driver.py: EAI 移动平台的驱动程序

launch:

demo.launch: 用于启动 EAI 移动平台驱动的启动文件

config:启动文件中包含的配置文件，

my\_dashgo\_params.yaml:用于配置和移动平台相关的一些参数

yocs\_velocity\_smoother.yaml:用于设置速度平滑

## 1.3dashgo\_tools

一系列控制移动平台的节点。

控制移动平台移动：roslaunch dashgo\_tools teleop\_twist\_keyboard.py

控制移动平台直线移动一定的距离（用于校准）：roslaunch dashgo\_tools check\_linear.py

控制移动平台旋转 360 度（用于校准）：roslaunch dashgo\_tools check\_angular.py

主要文件（或文件夹）：

scripts:一系列 python 程序，用于控制 EAI 移动平台

check\_angular.py: 用于转动校准

check\_linear.py: 用于移动校准

teleop\_twist\_keyboard.py: 用于通过键盘控制 EAI 移动平台

## 1.4dashgo\_nav

建图和导航

建图：roslaunch dashgo\_nav gmapping\_demo.launch

地图保存：

roslaunch map\_server map\_saver -f ~/catkin\_ws/src/dashgo/dashgo\_nav/maps/mymap

地图保存在 maps 文件下。

定位导航：roslaunch dashgo\_nav navigation\_demo.launch

主要文件（或文件夹）：

launch:包含建图和导航相关的启动文件

gmapping\_demo.launch:用于加载建图相关的包

navigation\_demo.launch: 用于加载导航相关的包

launch/include/odom:

amcl.launch.xml: amcl 包相关的参数配置

gmapping.launch: gmapping 包相关的参数配置

teb\_move\_base.launch: move\_base 包相关的参数配置

config/odom:包含启动文件需要包含的配置文件，主要用于规划器和代价地图配置

base\_global\_planner\_param.yaml: 用于全局规划器配置

costmap\_common\_params.yaml: 用于全局和局部代价地图的通用配置

global\_costmap\_params.yaml:用于全局代价地图的配置

local\_costmap\_params.yaml: 用于局部代价地图的配置  
teb\_local\_planner\_params.yaml: 用于局部规划期的配置

maps: 用于保存地图,  
mymap.pgm: 地图文件  
mymap.yaml: 地图相关的配置文件

## 1.5 dashgo\_rviz

定制 rviz 显示和导航相关的信息。  
roslaunch dashgo\_rviz view\_navigation.launch  
主要文件（或文件夹）：  
launch:  
    view\_navigation.launch: 导航时用于可视化的启动文件  
rviz:  
    view\_navigation.rviz: 和 rviz 相关的配置文件

## 1.6 navigation\_task

定制化导航任务。  
主要文件（或文件夹）：  
src: 包含执行导航任务相关的源文件 navigation\_node.cpp  
launch: 包含建图和导航相关的启动文件  
    gmapping\_start.launch: 用于加载建图相关的包  
    gmapping-start.sh: 用于建图的脚本文件  
    gmapping-save.sh: 用于保存地图的脚本文件  
    navigation\_imu.launch: 用于加载导航相关的包  
    navigation-start.sh: 用于导航的脚本文件

# 2 IMU 相关

## 2.1 jy901b\_imu

加载 IMU 驱动  
roslaunch jy901b\_imu demo.launch  
主要文件（或文件夹）：  
src: 包含驱动程序相关的源文件  
    transport\_serial.cpp : 串口配置  
    decoder.cpp : 编解码  
    jy901b.cpp : 获取 imu 数据  
    jy901b\_ros.cpp : imu 数据发布  
    main.cpp  
include: 包含驱动程序相关的头文件  
    transport.h   transport\_serial.h  
    decoder.h    jy901b.h           jy901b\_ros.h  
launch:  
    demo.launch: 驱动节点相关的启动文件

## 3 激光雷达相关

### 3.1 rplidar\_ros

加载激光雷达驱动

`roslaunch rplidar_ros rplidar.launch`

主要文件（或文件夹）：

src: 包含驱动程序相关的源文件

sdk: 激光雷达相关的软件开发包

launch:

    rplidar.launch: 驱动节点相关的启动文件

rviz: 包含和 rviz 相关的配置文件

## 三、开发说明

### 1 相关包的主要参数配置

#### 1.1 dashgo\_driver

相关参数设置在 config 文件夹中的 my\_dashgo\_params.yaml 和 yocs\_velocity\_smoother.yaml 中。

（1）my\_dashgo\_params.yaml: EAI 移动平台相关的参数设置。

EAI 移动平台对应的端口号     port: /dev/dashgo

EAI 移动平台上的坐标系名称   base\_frame: base\_footprint

移动平台相关的尺寸参数，需要校准

    轮径     wheel\_diameter: 0.125

    轮间距   wheel\_track: 0.348

              gear\_reduction: 1.0

              motors\_reversed: False

（2）yocs\_velocity\_smoother.yaml: 速度平滑相关的参数设置

速度限制     speed\_lim\_v: 0.6

              speed\_lim\_w: 1.5

加速度限制   accel\_lim\_v: 0.5

              accel\_lim\_w: 0.8

#### 1.2 jy901b\_imu

相关参数设置在 launch 文件夹中的 demo.launch 中。

IMU 对应的端口号     port: /dev/imu

是否选择调试模式   debug\_mode: false

角度补偿值         yaw\_correction: -0.0044

#### 1.3 rplidar\_ros

相关参数配置在 rplidar.launch 中。

激光雷达对应的端口号	serial_port: /dev/flashlidar
波特率	serial_baudrate: 115200
激光雷达的坐标系名称	frame_id: laser_frame
激光雷达是否倒置	inverted: true
是否角度补偿	angle_compensate: true

## 1.4 dashgo\_nav

(1) 建图相关的节点配置文件在 gmapping\_demo.launch 中  
加载移动平台驱动节点:

```
<include file="$(find dashgo_driver)/launch/demo.launch"/>
```

加载激光雷达驱动节点

```
<node name="rplidarNode"    pkg="rplidar_ros"    type="rplidarNode" output="screen">
  <param name="serial_port"      type="string" value="/dev/flashlidar"/>
  <param name="serial_baudrate"  type="int"    value="115200"/>
  <param name="frame_id"         type="string" value="laser_frame"/>
  <param name="inverted"         type="bool"   value="true"/>
  <param name="angle_compensate" type="bool"   value="true"/>
</node>
```

加载 EAI 移动平台模型

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

发布坐标系变换关系 (雷达相对 EAI 移动平台的关系, 需要安装后标定)

```
<node pkg="tf" type="static_transform_publisher" name="base_link_to_laser4"
args="0.31 0.0 0.24 -1.53 0 0 /base_footprint /laser_frame 40" />
```

加载建图相关的节点

```
<include file="$(find dashgo_nav)/launch/include/odom/gmapping.launch"/>
```

加载 move\_base 相关的节点

```
<include file="$(find dashgo_nav)/launch/include/odom/teb_move_base.launch"/>
```

建图相关的参数配置在 dashgo\_nav/launch/include/odom/gmapping.launch 中  
(slam\_gmapping 节点的相关参数配置):

坐标系设置 base\_frame: base\_footprint

odom\_frame: odom

地图更新时间设置 map\_update\_interval: 1

激光雷达的最大可用范围 maxUrange: 4.9

传感器的最大范围 maxRange: 5.0

请设置 maxUrange <真实传感器的最大范围<= maxRange。

kernelSize: 3

最小匹配得分 minimumScore: 30



决定了你对激光的一个置信度，越高说明你对激光匹配算法的要求越高，激光的匹配也越容易失败而转去使用里程计数据，而设的太低又会使地图中出现大量噪声。

滤波器的粒子数量      Particles: 80

地图初始尺寸      xmin: -5.0  
                     ymin:-6.0  
                     xmax:5.0  
                     ymax:4.0

不一定为最终的尺寸，与建立地图的环境有关。

地图分辨率    delta:0.01

(2) 定位导航相关的节点配置文件在 navigation\_demo.launch

加载移动平台驱动节点：

```
<include file="$(find dashgo_driver)/launch/demo.launch"/>
```

加载激光雷达驱动节点

```
<node name="rplidarNode"    pkg="rplidar_ros"    type="rplidarNode" output="screen">
  <param name="serial_port"            type="string" value="/dev/flashlidar"/>
  <param name="serial_baudrate"        type="int"     value="115200"/>
  <param name="frame_id"                type="string" value="laser_frame"/>
  <param name="inverted"                type="bool"    value="true"/>
  <param name="angle_compensate"        type="bool"    value="true"/>
</node>
```

加载 IMU 驱动节点

```
<node    pkg="jy901b_imu"        type="jy901b_imu_node"        name="jy901b_imu_node"
required="true" output="screen">
  <param name="port"                type="string" value="/dev/imu"/>
  <param name="debug_mode"          type="bool"     value="false"/>
  <param name="yaw_correction"        type="double" value="-0.0044"/>
</node>
```

发布坐标系变换关系（雷达相对 EAI 移动平台的关系，需要安装后标定）

```
<node pkg="tf" type="static_transform_publisher" name="base_link_to_laser4"
args="0.31 0.0 0.24 -1.53 0 0 /base_footprint /laser_frame 40" />
```

加载 maps 文件夹下的地图

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

启动定位相关的节点

```
<include file="$(find dashgo_nav)/launch/include/odom/amcl.launch.xml" />
```

启动 move\_base 相关的节点

```
<include file="$(find dashgo_nav)/launch/include/odom/teb_move_base.launch"/>
```

定位相关的参数配置 dashgo\_nav/launch/include/odom/amcl.launch.xml 中（amcl 节点的相关参数配置）：

```
里程模型类型（差分）      odom_model_type: diff
用于可视化的最大更新频率  gui_publish_rate: 5.0
考虑到的最大激光扫描范围  laser_max_range: 6.0
滤波器粒子数设置          min_particles: 2000
                           max_particles: 5000
执行滤波器更新需要的移动  update_min_d: 0.25
执行滤波器更新需要的转动  update_min_a: 0.2
```

```
坐标系设置  odom_frame_id: odom
             base_frame_id: base_footprint
             global_frame_id: map
此前的变换在未来多长时间内可用，时延  transform_tolerance: 1.0
初始位置  initial_pose_x: 0.0
           initial_pose_y: 0.0
           initial_pose_a: 0.0
```

move\_base 相关的参数配置在 dashgo\_nav/launch/include/odom/teb\_move\_base.launch 中（move\_base 节点配置）：

加载 costmap 配置

```
<rosparam file="$(find dashgo_nav)/config/odom/costmap_common_params.yaml"
command="load" ns="global_costmap" />
```

```
<rosparam file="$(find dashgo_nav)/config/odom/costmap_common_params.yaml"
command="load" ns="local_costmap" />
```

```
<rosparam file="$(find dashgo_nav)/config/odom/local_costmap_params.yaml"
command="load" />
```

```
<rosparam file="$(find dashgo_nav)/config/odom/global_costmap_params.yaml"
command="load" /
```

加载全局规划器配置

```
<rosparam file="$(find dashgo_nav)/config/odom/base_global_planner_param.yaml"
command="load" />
```

加载局部规划器配置

```
<rosparam file="$(find dashgo_nav)/config/odom/teb_local_planner_params.yaml"
command="load" />
```

全局规划器选择

```
<param name="base_global_planner" value="global_planner/GlobalPlanner"/>
```

局部规划器选择

```
<param name="base_local_planner" value="teb_local_planner/TebLocalPlannerROS" />
```

全局和局部代价地图相同部分配置在 dashgo\_nav/config/odom/costmap\_common\_params.yaml 中:

```
#robot_radius: 0.20 (圆形)
//移动平台尺寸设置, 多边形
footprint: [[0.35, 0.24], [-0.28, 0.24], [-0.28, -0.24], [0.35, -0.24]]

obstacle_layer:
  enabled: true
  obstacle_range: 4.0    距离机器人的最大默认距离
  raytrace_range: 5.0    扫描范围
  inflation_radius: 0.10  膨胀半径
  observation_sources: laser_scan_sensor  传感器类型选择
  track_unknown_space: true
  laser_scan_sensor: {data_type: LaserScan, topic: /scan, marking: true, clearing: true,
expected_update_rate: 0}

track_unknown_space: true

inflation_layer:
  enabled: true
  inflation_radius: 0.10  膨胀半径
```

局部代价地图配置在 dashgo\_nav/config/odom/local\_costmap\_params.yaml 中

```
local_costmap:
  global_frame: /odom      坐标系设置
  robot_base_frame: /base_footprint  坐标系设置
  update_frequency: 7.0    地图更新频率
  publish_frequency: 1.0   可视化相关的更新频率
  static_map: false        是否使用静态地图
  rolling_window: true      是否使用滑动窗口版本的 costmap。如果 static_map 参
数设置为 true, 则此参数必须设置为 false。
  width: 5.0               滑动窗口尺寸
  height: 5.0              滑动窗口尺寸
  resolution: 0.01         地图的分辨率
  transform_tolerance: 2.0  此前的变换在未来多长时间内可用, 时延
  map_type: costmap        地图类型
```

全局代价地图配置在 dashgo\_nav/config/odom/global\_costmap\_params.yaml 文件中

```
global_costmap:
  global_frame: /map      坐标系设置
  robot_base_frame: /base_footprint  坐标系设置
```

update_frequency: 1.0	地图更新频率
publish_frequency: 0	可视化相关的更新频率
static_map: true	是否使用静态地图
rolling_window: false	是否使用滑动窗口版本的 costmap。如果 static_map 参数设置为 true，则此参数必须设置为 false。
resolution: 0.01	地图的分辨率
transform_tolerance: 1.0	此前的变换在未来多长时间内可用，时延
map_type: costmap	地图类型

局部规划器配置在 dashgo\_nav/config/odom/teb\_local\_planner\_params.yaml 中

tebLocalPlannerROS:

odom_topic: odom	里程计相关的主题
map_frame: /odom	地图坐标系

dt\_ref: 0.40      期望轨迹的时间分辨率

global\_plan\_overwrite\_orientation: True 覆盖由全局规划器提供的局部子目标的方向(因为它们通常只提供 2D 路径)

max_vel_x: 0.55	最大移动速度
max_vel_x_backwards: 0.55	最大后退速度
max_vel_theta: 0.6	最大转动速度
acc_lim_x: 0.4	最大移动加速度
acc_lim_theta: 0.5	最大转动加速度
min_turning_radius: 0	最小转动半径

footprint\_model: # types: "point", "circular", "two\_circles", "line", "polygon"

types: "line"

line\_start: [-0.06, 0.0]

line\_end: [0.11, 0.0]

#radius: 0.20 # for type "circular"

机器人模型,和尺寸相关

# GoalTolerance

xy_goal_tolerance: 0.04	允许的最大距离误差
yaw_goal_tolerance: 0.04	允许的最大角度误差
free_goal_vel: False	去除目标速度约束，使机器人以最大速度到达目标

# Obstacles

#min\_obstacle\_dist: 0.25

min_obstacle_dist: 0.34	到障碍物的最小期望距离
include_costmap_obstacles: True	指定是否考虑局部代价地图的障碍物
costmap_obstacles_behind_robot_dist: 1.0	考虑规划时，限制占用的局部代价地图障碍
obstacle_poses_affected: 7	每个障碍物位置被附加到轨迹上最接近的姿势以保持距离。

也可以考虑临近的障碍物。

## 1.5 navigation\_task

定制化导航任务

navigation\_imu.launch

```
<launch>
```

导航相关节点

```
<include file="$(find dashgo_nav)/launch/navigation_demo.launch"/>
```

定制化导航任务节点

```
<node name="navigation_task" pkg="navigation_task" type="navigation_node"
```

```
output="screen" >
```

```
<param name="debug_mode" type="bool" value="true"/>
```

```
<param name="angle_adjust" type="bool" value="true"/>
```

```
<param name="angle_base" type="double" value="1.8"/>
```

```
</node>
```

```
</launch>
```

## 2 导航节点

由 *navigation\_task* 包下的源文件 *navigation\_node.cpp* 编译生成。下面为 *navigation\_node.cpp* 中的相关代码片段。

### 2.1 程序流程

- (1)读取位置信息（导航目标点）；
- (2)节点初始化；
- (3)发布消息和订阅消息设置；
- (4)sevice 客户端设置；
- (5)任务执行部分，循环等待任务命令：

id=2,data[0]=0: 回起点；

id=2,data[0]=1: 到达柜子第 1 列；

id=2,data[0]=2: 到达柜子第 2 列；

id=2,data[0]=3: 到达柜子第 3 列；

id=2,data[0]=4: 到达释放位置；

id=2,data[0]=5: 前进调整；

id=2,data[0]=6: 后退调整；

#### 2.1.1 “回起点”任务流程

- (1)接收到命令后回复；
- (2)判断是否在起点位置，若是执行步骤(3),否则执行步骤(4)；
- (3)任务完成后回复；
- (4)后退；测量 x,y 方向的距离；计算相对起点的角度和距离；先旋转计算的角度，在后退计算的距离；任务完成后回复。

没有使用 move\_base。

```
if ((2 == sub_msg.id) && (0 == sub_msg.data[0]) )  
{
```

```
    ROS_INFO("Need go to start positon"); //起点
```

```

pub_msg.id = 2;
pub_msg.data[0] = 14;
info_pub.publish(pub_msg);
//执行任务
if (false == is_start_pos)
{
    //通过激光雷达信息定位前方距离
    distance_adjust(twist_pub,-0.45,true); //退回
    laser_scan_distance = get_laser_length(1);
    float y_distance = 5.4 - 0.31 - 0.64 - laser_scan_distance;
    float x_distance = 5.4 - 0.24 - 0.40 - LaserScan_Y_Distance();
    float sita = atan(x_distance/y_distance)*180/3.1415926;

    ROS_INFO("x          Postion:%03f,y          Position:%0.3f,angle:%0.3f",
x_distance,y_distance,sita);
    orientation_adjust(twist_pub,-90 +sita );
    float back_distance = sqrt(x_distance*x_distance + y_distance*y_distance);
    move_distance(twist_pub,back_distance,-0.55,0.5);

    ROS_INFO("succeeded to move start position");
    //feedback
    pub_msg.id = 2;
    pub_msg.data[0] = 15;
    info_pub.publish(pub_msg);
    is_start_pos  = true;
}
else
{
    pub_msg.id = 2;
    pub_msg.data[0] = 15;
    info_pub.publish(pub_msg);
    is_start_pos  = true;
}
}

```

### 2.1.2 “到达柜子第 1 列” 任务流程

- (1)获取货架行列信息;
- (2)接收到命令后回复;
- (3)将目标位置发送到 move\_base 执行;
- (4)判断 move\_base 是否执行成功,若是执行步骤(5),否则执行步骤(6);
- (5)测量角度并将移动平台调整到 0 度,任务完成后回复;
- (6)打印失败提示。

```

if ((2 == sub_msg.id) && (1 == sub_msg.data[0]) )
{
    row_position = sub_msg.data[1];

```

```

column_position = sub_msg.data[0];
ROS_INFO("Need go to Postion1,1");
pub_msg.id = 2;
pub_msg.data[0] = 14;
info_pub.publish(pub_msg);
//货架第一列
goal.target_pose.header.frame_id = "map";
goal.target_pose.header.stamp = ros::Time::now();
if ( 1 == row_position )
{
    goal.target_pose.pose.position.x = grasp_pos1[0] - 0.10;
}
else
{
    goal.target_pose.pose.position.x = grasp_pos1[0];
}
goal.target_pose.pose.position.y = grasp_pos1[1];
goal.target_pose.pose.orientation.z = grasp_pos1[2];
goal.target_pose.pose.orientation.w = grasp_pos1[3];
ROS_INFO("Sending goal");
ac.sendGoal(goal);
ac.waitForResult();

if(ac.getState() == actionlib::SimpleClientGoalState::SUCCEEDED)
{
    if ( angle_adjust == true)
    {
        laser_scan_angle = lineFit(1,angle_base);
        orientation_adjust(twist_pub,0.0,laser_scan_angle);
    }

    ROS_INFO("succeeded to move pick position 1");
    //反馈
    pub_msg.id = 2;
    pub_msg.data[0] = 15;
    info_pub.publish(pub_msg);
    is_start_pos = false;
}
else

```

```

    {
        ROS_INFO("failed to move pick position 1 for some reason");
    }
}

```

### 2.1.3 “到达柜子第 2 列” 任务流程

流程和“到达柜子第 1 列”的任务流程一致，代码在加黑处由略微不同，主要和列信息有关。

### 2.1.4 “到达柜子第 3 列” 任务流程

流程和“到达柜子第 1 列”的任务流程一致，代码在加黑处由略微不同，主要和列信息有关。

### 2.1.5 “到达释放位置” 任务流程

- (1)接收到命令后回复;
- (2)imu 角度校正为 0;
- (3)旋转略小于-90 度;
- (4)将目标位置发送到 move\_base 执行;
- (5)判断 move\_base 是否执行成功，若是执行步骤(6),否则执行步骤(7);
- (6)测量角度并将移动平台调整到-90 度，向前移动 0.45 米，任务完成后回复;
- (7)打印失败提示。

```

if ((2 == sub_msg.id) && (4 == sub_msg.data[0]))
{
    ROS_INFO("Need go to Postion2");
    pub_msg.id = 2;
    pub_msg.data[0] = 14;
    info_pub.publish(pub_msg);
    //角度校 0， 去掉机械臂抓取过程中振动产生的干扰
    if (angle_to_zero_client.call(srv))
    {
        ROS_INFO("succeeded to call service");
    }
    else
    {
        ROS_ERROR("Failed to call service");
    }
    //目标释放
    //抓取位置不变，先调整姿态
    //旋转-90 度
    rotation_90deg(twist_pub);
    //到达一个临时位置，X 和姿态与目标相同，Y 方向有偏差
    goal.target_pose.header.frame_id = "map";
    goal.target_pose.header.stamp = ros::Time::now();

    goal.target_pose.pose.position.x = release_pos[0];
    goal.target_pose.pose.position.y = release_pos[1];
}

```



```

goal.target_pose.pose.orientation.z = release_pos[2];
goal.target_pose.pose.orientation.w = release_pos[3];

ROS_INFO("Sending goal");
ac.sendGoal(goal);
ac.waitForResult();

if(ac.getState() == actionlib::SimpleClientGoalState::SUCCEEDED)
{

    //到达目标位置向前移动 66cm
    if ( angle_adjust == true)
    {
        laser_scan_angle = lineFit(1,angle_base);
        orientation_adjust(twist_pub,-90, -90 + laser_scan_angle);
    }
    distance_adjust(twist_pub,0.45,true);
    ROS_INFO("succeeded to move place position");
    //反馈
    pub_msg.id = 2;
    pub_msg.data[0] = 15;
    info_pub.publish(pub_msg);
    pub_msg.id = 2;
    pub_msg.data[0] = 16;
    info_pub.publish(pub_msg);
    is_start_pos = false;
}
else
{
    ROS_INFO("failed to move place position for some reason");
}
}

```

### 2.1.6 “前进调整” 任务流程

- (1)接收到命令后回复;
- (2)慢速前进指定的距离;
- (3)测量角度并调整到 0 度;
- (4)任务完成后回复,包含实际移动距离与目标移动距离的差值。

```

if ((2 == sub_msg.id) && (5 == sub_msg.data[0]) )
{
    //接收的数据以厘米为单位,
    float distance = (float)sub_msg.data[1]/100;

    ROS_INFO("move forward %d",sub_msg.data[1]);
}

```

```

pub_msg.id = 2;
pub_msg.data[0] = 7;
info_pub.publish(pub_msg);

//move forward
double delta_distance = distance_adjust(twist_pub,distance);

laser_scan_angle = lineFit(column_position,angle_base);
orientation_adjust(twist_pub,0.0,laser_scan_angle);

//laser_scan_distance = get_laser_length(column_position);
//feedback
unsigned int move_distance= (unsigned int)(abs(delta_distance)*100);

pub_msg.id = 2;
pub_msg.data[0] = 8;
if (delta_distance >0)
    pub_msg.data[1] = 1;
else
    pub_msg.data[1] = 2;
pub_msg.data[2] = move_distance;
info_pub.publish(pub_msg);
is_start_pos = false;
}

```

### 2.1.7 “后退调整” 任务流程

- (1)接收到命令后回复;
- (2)慢速后退指定的距离;
- (3)任务完成后回复,包含实际移动距离与目标移动距离的差值。

```
if ((2 == sub_msg.id) && (6 == sub_msg.data[0]) )
```

```

{
    //接收的数据以厘米为单位
    float distance = 0 - (float)sub_msg.data[1]/100;
    ROS_INFO("move back %d",sub_msg.data[1]);
    pub_msg.id = 2;
    pub_msg.data[0] = 7;
    info_pub.publish(pub_msg);

    //move back
    double delta_distance = distance_adjust(twist_pub,distance);

    //feedback
    unsigned int move_distance= (unsigned int)(abs(delta_distance)*100);
    pub_msg.id = 2;

```

```

pub_msg.data[0] = 8;

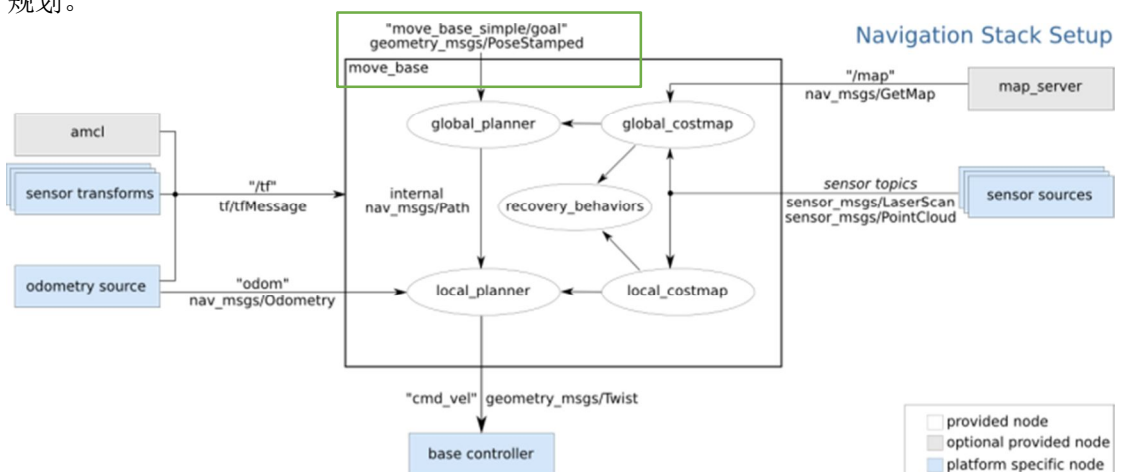
if (delta_distance > 0)
    pub_msg.data[1] = 1;
else
    pub_msg.data[1] = 2;
pub_msg.data[2] = move_distance;
info_pub.publish(pub_msg);
is_start_pos = false;
}
loop_rate.sleep();
++count;
}

```

## 2.2 目标点导航

使用 action 通信机制，可以在远程调用服务器上某程序后，转而执行其他的程序，周期性的获得服务器上的信息，或者取消发送到服务器上的某远程调用，机制灵活，异步通信。

需要定义客户端，通过客户端向服务器端发送导航请求，并由 move\_base 按照导航信息进行规划。



(1) 客户端定义:

```

typedef          actionlib::SimpleActionClient<move_base_msgs::MoveBaseAction>
MoveBaseClient;
MoveBaseClient *acp;

```

(2) 等待服务器端启:

```

MoveBaseClient ac("move_base", true);
acp = &ac;
while(!ac.waitForServer(ros::Duration(5.0))){
    ROS_INFO("Waiting for the move_base action server to come up");
}

```

(3) 设置目标点位置信息 (位置和姿态):

```

move_base_msgs::MoveBaseGoal goal;

```

```

goal.target_pose.header.frame_id = "map";
goal.target_pose.header.stamp = ros::Time::now();
goal.target_pose.pose.position.x = grasp_pos1[0];
goal.target_pose.pose.position.y = grasp_pos1[1];
goal.target_pose.pose.orientation.z = grasp_pos1[2];
goal.target_pose.pose.orientation.w = grasp_pos1[3];

```

(4)发送目标:

```
ac.sendGoal(goal);
```

(5)等待执行:

```
ac.waitForResult();
```

(6)执行结果判断:

```

if(ac.getState() == actionlib::SimpleClientGoalState::SUCCEEDED)
{

}

```

位置导航代码实例（已经实现步骤（1）、（2）的基础上）:

```
move_base_msgs::MoveBaseGoal goal;
```

```

goal.target_pose.header.frame_id = "map";
goal.target_pose.header.stamp = ros::Time::now();
goal.target_pose.pose.position.x = grasp_pos3[0];
goal.target_pose.pose.position.y = grasp_pos3[1];
goal.target_pose.pose.orientation.z = grasp_pos3[2];
goal.target_pose.pose.orientation.w = grasp_pos3[3];

```

```
ROS_INFO("Sending goal");
```

```
ac.sendGoal(goal);
```

```
ac.waitForResult();
```

```

if(ac.getState() == actionlib::SimpleClientGoalState::SUCCEEDED)
{

```

```
    if ( angle_adjust == true)
```

```
    {
```

```
        laser_scan_angle = lineFit(3,angle_base);
```

```
        orientation_adjust(twist_pub,0.0,laser_scan_angle);
```

```
        //orientation_adjust(twist_pub,0.0);
```

```
    }
```

```
    //反馈
```

```
    pub_msg.id = 2;
```

```
    pub_msg.data[0] = 15;
```

```
    info_pub.publish(pub_msg);
```

```
    is_start_pos = false;
```

```
}
```

```
else
```

```

    {
        ROS_INFO("failed to move pick position 3 for some reason");
    }
}

```

## 2.3 读取位置信息

读取/home/robot 目录下的 position.txt 文件。

每一行对应一个目标点的位置和姿态。第 1 列和第 2 列分别为 x 和 y 位置，第 3 列和第 4 列为姿态数据，分别为四元数中的 Z 和 W（2D 下，X，Y 为 0 省略）。

```

double start_pos[4];
double grasp_pos1[4],grasp_pos2[4],grasp_pos3[4];
double release_pos[4];
double target_pos[20];
int pos_num = 0;
bool is_start_pos = true;
//read the position of navigation from the txt file
ifstream Object_position_file("/home/robot/position.txt");
string temp;
if(!Object_position_file.is_open())
{
    cout<<"Failed to open  position file"<<endl;
    return 0;
}

while(getline(Object_position_file,temp))
{
    sscanf(temp.c_str(), "%lf %lf %lf %lf", &target_pos[4*pos_num], &target_pos[4*pos_num+1], &target_pos[4*pos_num+2], &target_pos[4*pos_num+3]);
    pos_num++;
}
for(int i=0; i<pos_num; i++)

    cout<<target_pos[4*i]<<"\t"<<target_pos[4*i+1]<<"\t"<<target_pos[4*i+2]<<"\t"<<target_pos[4*i+3]<<endl;
    Object_position_file.close();
    for (int i = 0; i < 4; i++)
    {
        start_pos[i]    = target_pos[i];
        grasp_pos1[i]   = target_pos[4+i];
        grasp_pos2[i]   = target_pos[8+i];
        grasp_pos3[i]   = target_pos[12+i];
        release_pos[i]  = target_pos[16+i];
    }
    cout<<grasp_pos2[0]<<"\t"<<grasp_pos2[1]<<"\t"<<grasp_pos2[2]<<"\t"<<grasp_pos2[3]<
<endl;

```

## 2.4 发布的消息

(1) 任务相关消息:

```
id_data_msgs::ID_Data pub_msg;
```

```
ros::Publisher info_pub = n.advertise<id_data_msgs::ID_Data>("/notice",10);
```

(2) 速度控制消息:

```
geometry_msgs::Twist speed_msgs;
```

```
ros::Publisher twist_pub = n.advertise<geometry_msgs::Twist>("/cmd_vel",10);
```

## 2.5 订阅的消息

(1) 任务相关消息:

```
id_data_msgs::ID_Data sub_msg;
```

```
ros::Subscriber sub = n.subscribe("/notice", 100, infoCallback);
```

(2) imu 角度消息:

```
std_msgs::Float32 imu_angle_msgs;
```

```
ros::Subscriber imu_angle_sub
```

```
n.subscribe("/jy901b_imu/imu_angle",5,imuAngleCallback);
```

(3) 里程消息:

```
nav_msgs::Odometry odom_msg;
```

```
ros::Subscriber odom_data = n.subscribe("/odom",10,odomCallback);
```

(4) 激光雷达扫描数据:

```
ros::Subscriber laser_scan_data = n.subscribe("/scan",1,laser_scanCallback);
```

## 2.6 消息回调函数

(1)任务相关消息:

接收 id, 以及包含相关的控制命令的 data 数组。

```
void infoCallback(const id_data_msgs::ID_Data::ConstPtr& msg)
```

```
{
    sub_msg.id = msg->id;
    sub_msg.data[0] = msg->data[0];
    if ((msg->id == 2) && (msg->data[0] < 7) && (msg->data[0] > 0))
    {
        sub_msg.data[1] = msg->data[1];
    } else if ((msg->id == 5) && (msg->data[0] == 14) ) {
        acp->cancelGoal();    //取消导航任务
    }
}
```

(2)imu 角度消息:

只接收航向角相关的信息。

```
void imuAngleCallback(const std_msgs::Float32::ConstPtr& msg)
{
    imu_angle_msgs.data = msg->data;
    if ((imu_angle_msgs.data >= -180) && (imu_angle_msgs.data < -135))
    {
        imu_angle_msgs.data += 360;
    }
}
```

(3)里程消息:

只接收 x,y 位置以及 x 方向上的线速度。

```
void odomCallback(const nav_msgs::Odometry::ConstPtr& msg )
{
    odom_msg.pose.pose.position.x = msg->pose.pose.position.x;
    odom_msg.pose.pose.position.y = msg->pose.pose.position.y;
    odom_msg.twist.twist.linear.x = msg->twist.twist.linear.x;
}
```

(4) 激光雷达扫描数据;

获取的数据包括两部分。

第一部分为移动平台正前方的 61 个数据点，以移动平台的 x 轴为中心，左右对称。其用途为：测量角度，计算障碍物距离移动平台的距离（移动平台 X 轴方向）。

第二部分为移动平台左侧的 20 个数据点，其用途为：计算障碍物距离移动平台的距离（移动平台 Y 轴方向）。

```
void laser_scanCallback(const sensor_msgs::LaserScan::ConstPtr& msg )
{
    angle_min = msg->angle_min;
    angle_increment = msg->angle_increment;
    //获取 X 方向和角度的测距信息
    for (int i = 0;i< 61;i++)
    {
        if (msg->intensities[i+237] > 1)
        {
            ranges[i] = msg->ranges[i+237];
        }
        else
        {
            ranges[i] = 0;
        }
    }
}
```

```

//获取 Y 方向距离的测距信息
for (int i = 0;i< 20;i++)
{
    if (msg->intensities[i+340] > 1)
    {
        ranges_y[i] = msg->ranges[i+340] * sin(angle_min+angle_increment*(i+340)
        - 1.53);
    }
    else
    {
        ranges_y[i] = 0;
    }
}
}

```

## 2.7 service 服务

客户端定义，用于 imu 的航向角重新置 0。

```

ros::ServiceClient          angle_to_zero_client          =
n.serviceClient<std_srvs::Empty>("/jy901b_imu/set_zero_orientation");
std_srvs::Empty srv;

```

开始前往释放位置时调用

```

//角度校 0， 去掉机械臂抓取过程中振动产生的干扰
if (angle_to_zero_client.call(srv))
{
    ROS_INFO("succesed to call service");
}
else
{
    ROS_ERROR("Failed to call service");
}

```

## 2.8 参数设置

- (1)angle\_adjust: 是否启用角度调整;
- (2)debug\_mode: 是否启用调试模式;
- (3)angle\_base: 用于补偿激光雷达测得的角度，为移动平台与柜子平行时的测距角度的相反数。

```

bool angle_adjust = false;
bool debug_mode = false;
double angle_base = 0.0;
n.param<bool>("/navigation_task/angle_adjust", angle_adjust, false);
n.param<bool>("/navigation_task/debug_mode", debug_mode, false);

```



```
n.param<double>("/navigation_task/angle_base", angle_base, 0.0);
```

## 2.9 相关函数

(1) void rotation\_90deg(ros::Publisher& cmd)

顺时针旋转略大于 90 度（负值），角速度为-0.70rad/s

//旋转-90 度

```
void rotation_90deg(ros::Publisher& cmd)
```

```
{
    ROS_INFO("rotation -90degree");
    ros::Rate loop_rate(20);
    unsigned int count = 0;
    //延时 6.2 秒旋转-90 度
    while(ros::ok())
    {
        count++;
        speed_msgs.angular.z = -0.70;
        cmd.publish(speed_msgs);
        if (count >45)
            break;
        loop_rate.sleep();
    }
    count = 0;
    //延时 1s,角速度变为 0
    while(ros::ok())
    {
        count++;
        speed_msgs.angular.z = 0;
        cmd.publish(speed_msgs);
        if (count >2)
            break;
        loop_rate.sleep();
    }
}
```

(2) void orientation\_adjust(ros::Publisher& cmd,float target\_angle,float laser\_angle)

角度调整，调整之前，以激光雷达测量的角度校正 imu 的测量，调整过程中只使用 imu 的测量数据。

调整的角度大于 10 度时， -0.15rad/s 的角速度调整

调整的角度小于 10 度，大于 0.4 度时，以-0.07rad/s 的角速度调整

调整的角度小于-10 度时， 0.15rad/s 的角速度调整

调整的角度大于-10 度，小于-0.4 度时，以 0.07rad/s 的角速度调整

-0.4 到 0.4 之间不调整

```
void orientation_adjust(ros::Publisher& cmd,float target_angle,float laser_angle)
```

```

{
    ROS_INFO("angle adjust");
    ros::Rate loop_rate(60);
    unsigned int count = 0;
    bool is_start = true;
    //wait 1 second in order to receive imu sensor
    while(ros::ok())
    {
        count++;
        if (count > 5)
            break;
        loop_rate.sleep();
    }
    while(ros::ok())
    {
        count++;
        if (is_start == true)
        {
            ROS_INFO("imu_angle %0.3f,laser_angle %0.3f",imu_angle_msgs.data,laser_angle);
            target_angle = target_angle + (imu_angle_msgs.data - laser_angle);
            is_start = false;
        }
        if ((imu_angle_msgs.data - target_angle) > 10)
        {
            if (count % 3 == 0)
            {
                ROS_INFO("angle %0.3f",imu_angle_msgs.data);
            }
            speed_msgs.angular.z = -0.15;
            cmd.publish(speed_msgs);
        }
        else
        {
            break;
        }
        loop_rate.sleep();
    }
    while(ros::ok())
    {
        count++;
        if ((imu_angle_msgs.data - target_angle) > 0.4)
        {
            if (count % 3 == 0)
            {

```

```

        ROS_INFO("angle %f",imu_angle_msgs.data);
    }
    speed_msgs.angular.z = -0.07;
    cmd.publish(speed_msgs);
}
else
{
    if (((imu_angle_msgs.data - target_angle) > -0.4) && (imu_angle_msgs.data -
target_angle) < 0.4)
    {
        if ( count % 3 == 0)
        {
            ROS_INFO("angle %f",imu_angle_msgs.data);
        }
        speed_msgs.angular.z = 0;
        cmd.publish(speed_msgs);
        ROS_INFO("angle adjust 0");
    }
    break;
}
loop_rate.sleep();
}

while(ros::ok())
{
    count++;
    if ((imu_angle_msgs.data - target_angle) < -10)
    {
        if ( count % 3 == 0)
        {
            ROS_INFO("angle %f",imu_angle_msgs.data);
        }
        speed_msgs.angular.z = 0.15;
        cmd.publish(speed_msgs);
    }
    else
    {
        break;
    }
    loop_rate.sleep();
}
while(ros::ok())
{
    count++;

```

```

        if ((imu_angle_msgs.data - target_angle) < -0.4)
        {
            if ( count % 3 == 0)
            {
                ROS_INFO("angle %f",imu_angle_msgs.data);
            }
            speed_msgs.angular.z = 0.07;
            cmd.publish(speed_msgs);
        }
        else
        {
            if (((imu_angle_msgs.data - target_angle) > -0.4) && (imu_angle_msgs.data -
target_angle) < 0.4)
            {
                if ( count % 3 == 0)
                {
                    ROS_INFO("angle %f",imu_angle_msgs.data);
                }
                speed_msgs.angular.z = 0;
                cmd.publish(speed_msgs);
                ROS_INFO("angle adjust 0");
            }
            break;
        }
        loop_rate.sleep();
    }

    if (((imu_angle_msgs.data - target_angle) > -0.4) && (imu_angle_msgs.data -
target_angle) < 0.4)
    {
        if ( count % 3 == 0)
        {
            ROS_INFO("angle %f",imu_angle_msgs.data);
        }
        speed_msgs.angular.z = 0;
        cmd.publish(speed_msgs);
        ROS_INFO("angle adjust 0");
    }
}

```

(3) void move\_distance(ros::Publisher& cmd,float distance,float speed = 0.55,float acc = 0.5)

快速移动较大的距离，没有使用里程计反馈，通过  $0.5 \times \text{加速度} \times \text{时间} \times \text{时间} = \text{距离}$  来计算移动时间。

void move\_distance(ros::Publisher& cmd,float distance,float speed = 0.55,float acc = 0.5)

```

{
    ros::Rate loop_rate(100);
    int count = 0;
    float acc_time = abs(speed)/acc;
    float time;
    float acc_distance = acc*acc_time * acc_time;    /*a*t2

    if (distance >= acc_distance)
        time = (acc_time + (distance - acc_distance)/abs(speed))*100;
    else
    {
        ime = sqrt(distance/acc)*100;
    }

    while(ros::ok())
    {
        if (count > time)
            break;
        speed_msgs.linear.x = speed;
        speed_msgs.angular.z = 0.0;
        cmd.publish(speed_msgs);
        count++;
        loop_rate.sleep();
    }
    count = 0;
    while(ros::ok())
    {
        if (count > acc_time)
            break;
        speed_msgs.linear.x = 0;
        speed_msgs.angular.z = 0.0;
        cmd.publish(speed_msgs);
        count++;
        loop_rate.sleep();
    }
}

```

(4) double distance\_adjust(ros::Publisher& cmd,float distance,bool fast\_mode = false)

慢速调整距离，根据里程计反馈来判断是否结束调整。

double distance\_adjust(ros::Publisher& cmd,float distance,bool fast\_mode = false)

```

{
    double start_position_x,start_position_y;
    double delta_x,delta_y;
    double adjust_factor;    //距离调整因子， .05m/s 停止移动距离大约 0.022m

```

```

double target_distance = abs(distance);
double delta_distance = 0.0;

//打滑检测
std_msgs::Float32 imu_base_angle_msgs;
std_msgs::Float32 imu_delta_angle_msgs;

if (target_distance < 0.05)
    adjust_factor = 2.22;
else
{
    if (fast_mode == true)
        adjust_factor = target_distance/(target_distance - 0.100);
    else
        adjust_factor = target_distance/(target_distance - 0.022);
}
ROS_INFO("distance adjust %0.3f", distance);
double current_distance = 0.0;
bool start_flag = false;
bool stop_flag = false;
ros::Rate loop_rate(60);
int count = 0;
while(ros::ok())
{
    while(ros::ok())
    {
        count++;
        if (count > 3)
            break;
        loop_rate.sleep();
    }
    if (!start_flag)
    {
        start_position_x = odom_msg.pose.pose.position.x;
        start_position_y = odom_msg.pose.pose.position.y;

        imu_base_angle_msgs.data = imu_angle_msgs.data;
        start_flag = true;
    }

    if ((sub_msg.id == 5) && (sub_msg.data[0] == 14) )
    {
        ROS_INFO("stop!!!!");
        stop_flag = true;
    }
}

```

```

    }

    if (stop_flag == true)
    {
        speed_msgs.linear.x = 0.0;
        speed_msgs.angular.z = 0.0;
        cmd.publish(speed_msgs);
    }
    if (stop_flag == false)
    {
        if (distance > 0.0)
        {
            if (fast_mode == true)
                speed_msgs.linear.x = 0.55;
            else
                speed_msgs.linear.x = 0.05;
        }
        else
        {
            if (fast_mode == true)
                speed_msgs.linear.x = -0.55;
            else
                speed_msgs.linear.x = -0.05;
        }

        imu_delta_angle_msgs.data      =      abs(imu_base_angle_msgs.data      -
imu_angle_msgs.data);

        if ( imu_delta_angle_msgs.data > 2)
        {
            //ROS_INFO("delta_angle:%0.3lf",imu_delta_angle_msgs.data);
        }

        speed_msgs.angular.z = 0.0;
        delta_x = odom_msg.pose.pose.position.x - start_position_x;
        delta_y = odom_msg.pose.pose.position.y - start_position_y;
        current_distance = sqrt(delta_x*delta_x + delta_y*delta_y);
        //ROS_INFO("distance:%0.3lf",current_distance);
        if ((current_distance*adjust_factor) >= target_distance )
        {
            speed_msgs.linear.x = 0.0;
            cmd.publish(speed_msgs);
            count=0;
            while(ros::ok())

```

```

        {
            if      (odom_msg.twist.twist.linear.x      <      0.01      &&
odom_msg.twist.twist.linear.x > -0.01)
            {
                delta_x = odom_msg.pose.pose.position.x - start_position_x;
                delta_y = odom_msg.pose.pose.position.y - start_position_y;
                current_distance = sqrt(delta_x*delta_x + delta_y*delta_y);
                delta_distance = target_distance - current_distance;
                ROS_INFO("delta_distance%0.3f",delta_distance);
                return delta_distance;
            }
            loop_rate.sleep();
        }
        break;
    }
    else
    {
        cmd.publish(speed_msgs);
    }
}
loop_rate.sleep();
++count;
}
return 0.0;
}

```

(5) float lineFit(int column, float laser\_angle\_base)

通过最小二乘法拟合 31 个激光雷达测量数据相对于移动平台 y 轴的角度。

总共接收 61 个激光雷达测量数据，为避免柜子支撑腿的影响，根据货架的列信息，每次只使用最右侧、中间或最左侧的 31 个测量数据。

float lineFit(int column, float laser\_angle\_base)

```

{
    int count = 0;
    int value_count = 0;

    float A = 0.0;
    float B = 0.0;
    float C = 0.0;
    float D = 0.0;
    float E = 0.0;
    float F = 0.0;
    float a, b, temp = 0;
    float angle;

```



```

ros::Rate loop_rate(30);
while(ros::ok())
{
    count++;
    if (count > 5)
        break;
    loop_rate.sleep();
}
while(ros::ok())
{
    value_count = 0;
    for (int i = 0 + (column - 1) * 15; i < 31 + (column - 1) * 15; i++)
    {
        if (ranges[i] > 0.05)
        {
            position_x = ranges[i] * cos(angle_min + angle_increment * (i + 237) - 1.53);
            position_y = ranges[i] * sin(angle_min + angle_increment * (i + 237) - 1.53);

//ROS_INFO("index:%d,data:%0.3f,x:%0.3f,y:%0.3f",i+237,ranges[i],position_x,position_y);
            A += position_y * position_y;
            B += position_y;
            C += position_y * position_x;
            D += position_x;
            value_count++;
        }
    }
    // 执行拟合数据点 position_x[value_count] , position_y[value_count] , 数量
    value_count

    //以 position_y 为横轴, position_x 为纵轴

    // 计算斜率 a 和截距 b
    temp = (value_count * A - B * B);
    if( temp > 0.000001 || temp < -0.000001 )// 判断分母不为 0
    {
        a = (value_count * C - B * D) / temp;
        b = (A * D - B * C) / temp;
        angle = atan(a) * 180 / 3.1415926 + laser_angle_base;
        ROS_INFO("angle:%0.3f",angle);
        return angle;
    }
    else
    {
        a = 1;
        b = 0;
    }
}

```

```

        return laser_angle_base;

    }
    loop_rate.sleep();
}
return 0;
}

```

(6)float LaserScan\_Y\_Distance()

获取激光雷达距离左侧标志物的距离(移动平台 Y 方向)，Y 方向取均值。

//获取激光雷达测距信息 Y 方向， 距离左侧标志物的距离

```

float LaserScan_Y_Distance()
{
    int count = 0;
    int value_count = 0;

    float distance;
    float sum = 0;

    ros::Rate loop_rate(30);
    while(ros::ok())
    {
        count++;
        if (count > 5)
            break;
        loop_rate.sleep();
    }
    while(ros::ok())
    {
        value_count = 0;
        for (int i = 0; i < 20; i++)
        {
            if ( (ranges_y[i] > 0.20) && (ranges_y[i] < 5.0) )
            {
                sum += ranges_y[i];
                value_count++;
            }
        }
        distance = sum / value_count;
        if ( (distance > 0.2) && (distance < 5.0) )
        {
            ROS_INFO("Y_Distance:%0.3f", distance);
            break;
        }
    }
}

```

```

    loop_rate.sleep();
}
return distance;
}

```

(7)float get\_laser\_length(int column)

获取激光雷达距离前方标志物的距离(移动平台 X 方向), 为避免柜子支撑腿的影响, 根据货架的列信息, 每次只使用最右侧、中间或最左侧的 31 个测量数据, X 方向取均值。

//获取激光雷达测距信息 X 方向, 距离前方标志物的距离

```

float get_laser_length(int column)
{
    float sum = 0.0;
    float distance;
    int count = 0;
    int value_count = 0;
    ros::Rate loop_rate(30);
    while(ros::ok())
    {
        count++;
        if (count > 5)
            break;
        loop_rate.sleep();
    }
    while(ros::ok())
    {
        for (int i = 0 + (column - 1) * 15; i < 31 + (column - 1) * 15; i++)
        {
            if ( ranges[i] > 0.05)
            {
                position_x = ranges[i] * cos(angle_min + angle_increment * (i + 237) - 1.53);
                //ROS_INFO("data:%0.3f,position_x:%0.3f",ranges[i],position_x);
                value_count++;
                sum += position_x;
            }
        }
        if (value_count != 0)
        {
            distance = sum / value_count;
            if (distance > 0.15)
            {
                ROS_INFO("distance:%0.3f",distance);
                return distance;
            }
        }
    }
}

```

```

    }
    loop_rate.sleep();
}
return 0;
}

```

## 四、操作说明

### 1 硬件校准

#### 1.1 移动平台校准

参考 PS1000B 用户手册移动平台校准部分(P15)

(1) 首先加载 EAI 平台驱动:

```
roslaunch dashgo_driver demo.launch
```

(2) 控制移动平台直线移动一定的距离 (2 米, 用于校准):

```
roslaunch dashgo_tools check_linear.py
```

(3) 控制移动平台旋转 360 度 (用于校准):

```
roslaunch dashgo_tools check_angular.py
```

优先校准走直线, 这个误差达到要求后在校准转动角度。走直线只和轮子直径有关, 转动角度既和轮径有关, 还和轮间距有关。

校准走直线时, 实际运行超过要求运行的距离时, 调大轮子直径; 实际运行不足要求运行的距离时, 调小轮子直径。

校准转动 360 度时, 实际转动超过 360 度时, 调小轮子间距; 实际转动不足 360 度时, 调大轮子间距。

如果轮子直径和轮间距已明显高于轮子实际的直径和间距, 需要通过调整动力系数使运行达到精准。

误差应控制在 1%, 最终移动的距离和角度应以查看里程计数据为准, 并和测量的距离和角度比较。

相关参数设置在 config 文件夹中的 my\_dashgo\_params.yaml 中:

```

轮径      wheel_diameter: 0.125
轮间距    wheel_track: 0.348
动力系数  gear_reduction: 1.0

```

可以通过修改 dashgo/dashgo\_tools/scripts/ check\_linear.py 中的  
`self.test_distance = rospy.get_param('~test_distance',测量距离的值)`  
 把测量距离修改为其他值。

可以通过修改 dashgo/dashgo\_tools/scripts/check\_angular.py 中的  
`self.test_distance = rospy.get_param('~test_distance',测量角度的值)`  
 被测量角度修改为其他值, 但不能超过 360 度。

#### 1.2 IMU 校准:

此处校准只是 IMU 的角度校准。

在终端启动移动平台驱动: `roslaunch dashgo_driver demo.launch`

在终端启动 IMU 驱动: `roslaunch jy901b_imu demo.launch`

启动移动平台控制节点：`roslaunch dashgo_tools teleop_twist_keyboard.py`

控制移动平台转动 360 度，查看 imu 的角度。用 360 度减去 imu 的角度，将差值除以 360 度作为补偿值。

### 1.3 激光雷达校准

参考 PS1000B 用户手册激光雷达校准部分(P17)

(1) 运行建图节点

```
roslaunch dashgo_nav gmapping_demo.launch
```

(2) 运行 rviz 进行观察

```
roslaunch dashgo_rviz view_navigation.launch
```

调整的参数为 `gmapping_demo.launch` 中的：

```
<node pkg="tf" type="static_transform_publisher" name="base_link_to_laser4"
args="0.31 0.0 0.24 -1.53 0 0 /base_footprint /laser_frame 40" />
```

#### 1.3.1 调整的参数

在 `/base_footprint` 前面的 6 个参数，只需要调整前面 5 个参数，最后一个默认为 0.0 即可。

这 5 个参数均为相对于移动平台的坐标系来调整的。

前面 3 个分别表示激光雷达在 X、Y、Z 轴（右手定则）上距离 (0,0,0) 点的坐标位置，(0,0,0) 点是移动平台的坐标系原点，为其重心点。

后面 2 个分别表示沿着移动平台中心线（正前方与正后方的连线），第一个参数是左右方向上偏离中心线所在垂直平面的角度，第二个参数则是上下方向偏离中心线所在水平平面的角度，大小范围为 -3.1415926~3.1415926。

#### 1.3.2 前方方向校正

只需要修改第 4 个参数。

移动平台的前方有 3 个超声波模块，中间的超声波模块就是正前方。移动平台的后方有 1 个超声波模块。

一般需要借用较大的平整的挡板来确定激光雷达当前的方位。先将挡板横摆在移动平台正前方，挡板中心与移动平台正前方那个超声波模块成一条直线，挡板的面要与直线成 90 度角。

然后观察 rviz 中的显示，若与实际一致则方向正确，否则调整第 4 个参数。

#### 1.3.3 左右方向校正

只需修改第 5 个参数。

先将挡板横摆在移动平台的正左方，挡板中心与移动平台中心成一条线。然后在 rviz 中观察，是否与实际一致。若不一致，则修改第 5 个参数。若当前数值为 0.0，则修改成 3.1415926；若当前数值为 3.1415926，则修改成 0.0。

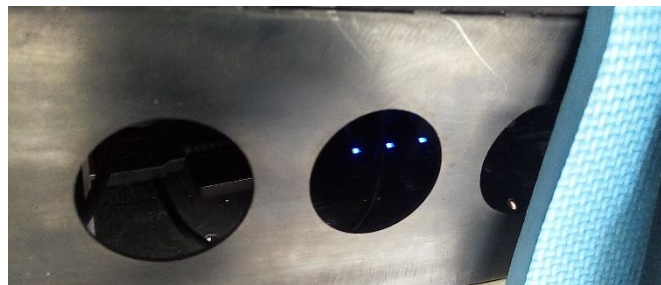
## 2 建图与导航

### 2.1 dashgo 启动前检查

(1) 检查 dashgo 供电电源是否开启，若开启，电压低于 12V 需要充电，运行电压最低不能低于 11V。



(2) 检查 HUB 上的设备是否开启，指示灯亮表示开启，右侧第一个对应移动平台 dashgo，右侧第二个对应 IMU。激光雷达在 NUC 主机上，Kinect 下方。



(3) 检查端口号是否正常，显示和下图一致为正常结果。

```
robot@robot-PC: ~  
robot@robot-PC:~$ ls /dev/dashgo  
/dev/dashgo  
robot@robot-PC:~$ ls /dev/flashlidar  
/dev/flashlidar  
robot@robot-PC:~$ ls /dev/imu  
/dev/imu  
robot@robot-PC:~$
```

## 2.2 端口连接问题

(1) dashgo 端口识别问题：ttyACM0 无法识别。

关闭移动平台电源按键，重启 dashgo 在 hub 上的对应端口，开启 dashgo 电源，还是无法识别，关闭其他节点重试以上步骤。

(2) IMU 和雷达端口已做端口绑定，不要调整相关的 USB 端口。

## 2.3 dashgo 控制

(1) 启动 dashgo 相关节点。

roslaunch dashgo\_driver\_demo.launch

```
/home/robot/catkin_ws/src/dashgo/dashgo_driver/launch/demo.launch http://localhost:11311
robot@robot-PC:~$ roslaunch dashgo_driver_demo.launch
... logging to /home/robot/.ros/log/87d2a0b6-0e02-11e7-9de1-b50b97a1861f/roslauch
ch-robot-PC-31007.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://robot-PC:44774/

SUMMARY
=====

PARAMETERS
* /dashgo_driver/Kd: 20
* /dashgo_driver/Ki: 0
* /dashgo_driver/Ko: 50
* /dashgo_driver/Kp: 50
* /dashgo_driver/accel_limit: 1.0
* /dashgo_driver/base_controller_rate: 10
* /dashgo_driver/base_frame: base_footprint
* /dashgo_driver/baud: 115200
* /dashgo_driver/encoder_resolution: 1200
* /dashgo_driver/gear_reduction: 1.0
* /dashgo_driver/motors_reversed: False
* /dashgo_driver/port: /dev/dashgo
* /dashgo_driver/rate: 50
* /dashgo_driver/sensorstate_rate: 10
* /dashgo_driver/timeout: 0.1
* /dashgo_driver/useImu: False
* /dashgo_driver/use_base_controller: True
* /dashgo_driver/wheel_diameter: 0.125
* /dashgo_driver/wheel_track: 0.348
* /rostdistro: indigo
* /rosversion: 1.11.20
* /use_sim_time: False
* /velocity_smoother/accel_lim_v: 0.2
* /velocity_smoother/accel_lim_w: 0.5
* /velocity_smoother/decel_factor: 1.0
* /velocity_smoother/frequency: 20.0
* /velocity_smoother/robot_feedback: 0
* /velocity_smoother/speed_lim_v: 0.4
```

正常启动应于下图显示类似：

```
/home/robot/catkin_ws/src/dashgo/dashgo_driver/launch/demo.launch http://localhost:11311
* /velocity_smoother/speed_lim_v: 0.4
* /velocity_smoother/speed_lim_w: 1.5

NODES
/
dashgo_driver (dashgo_driver/dashgo_driver.py)
nodelet_manager (nodelet/nodelet)
velocity_smoother (nodelet/nodelet)

auto-starting new master
process[master]: started with pid [31019]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 87d2a0b6-0e02-11e7-9de1-b50b97a1861f
process[rosout-1]: started with pid [31032]
started core service [/rosout]
process[dashgo_driver-2]: started with pid [31039]
process[nodelet_manager-3]: started with pid [31047]
process[velocity_smoother-4]: started with pid [31052]
[DEBUG] [WallTime: 1490078969.752738] init_node, name[/dashgo_driver], pid[31039]
[DEBUG] [WallTime: 1490078969.752968] binding to 0.0.0.0 0
[DEBUG] [WallTime: 1490078969.753109] bound to 0.0.0.0 33570
[DEBUG] [WallTime: 1490078969.753386] ... service URL is rosrpc://robot-PC:33570
[DEBUG] [WallTime: 1490078969.753537] [/dashgo_driver/get_loggers]: new Service
instance
[DEBUG] [WallTime: 1490078969.754601] ... service URL is rosrpc://robot-PC:33570
[DEBUG] [WallTime: 1490078969.754752] [/dashgo_driver/set_logger_level]: new Ser
vice instance
Connecting to Arduino on port /dev/dashgo ...
Connected at 115200
Arduino is ready.
[INFO] [WallTime: 1490078970.767577] Connected to Arduino on port /dev/dashgo at
115200 baud
Updating PID parameters
[INFO] [WallTime: 1490078970.790055] Started base controller for a base of 0.348
m wide with 1200 ticks per rev
[INFO] [WallTime: 1490078970.790337] Publishing odometry data at: 10.0 Hz using
base_footprint as base frame
[DEBUG] [WallTime: 1490078970.816590] connecting to robot-PC 50164
```



(2) 键盘控制 dashgo 移动。

roslaunch dashgo\_tools teleop\_twist\_keyboard.py

```
Terminal
robot@robot-PC: ~
robot@robot-PC:~$ roslaunch dashgo_tools teleop_twist_keyboard.py
the roslaunch view is empty: call 'sudo roslaunch init' and 'roslaunch update'

Reading from the keyboard and Publishing to Twist!
-----
Moving around:
  u    i    o
  j    k    l
  m    ,    .
-----
For Holonomic mode (strafing), hold down the shift key:
-----
  U    I    O
  J    K    L
  M    <    >
-----
t : up (+z)
b : down (-z)
anything else : stop
q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
CTRL-C to quit

currently:      speed 0.3      turn 0.6

[DEBUG] [WallTime: 1490078969.754601] ... service URL is rosrpc://robot-PC:33570
[DEBUG] [WallTime: 1490078969.754752] [/dashgo_driver/set_logger_level]: new Ser
vice instance
Connecting to Arduino on port /dev/dashgo ...
Connected at 115200
Arduino is ready.
[INFO] [WallTime: 1490078970.767577] Connected to Arduino on port /dev/dashgo at
115200 baud
Updating PID parameters
[INFO] [WallTime: 1490078970.790055] Started base controller for a base of 0.348
m wide with 1200 ticks per rev
[INFO] [WallTime: 1490078970.790337] Publishing odometry data at: 10.0 Hz using
base_footprint as base frame
[DEBUG] [WallTime: 1490078970.816590] connecting to robot-PC 50164
```

向前移动 I 向后移动 < 逆时针旋转 J 顺时针旋转 L

速度调整 q/z 线速度和角速度调整 w/x 线速度调整 e/c 角速度调整

```
Terminal
robot@robot-PC: ~
-----
  U    I    O
  J    K    L
  M    <    >
-----
t : up (+z)
b : down (-z)
anything else : stop
q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
CTRL-C to quit

currently:      speed 0.3      turn 0.6
```

## 2.4 建立环境地图

(1) 将 dashgo 摆正，初始位置做好标记，按照第一部分做好检查。





(2) 运行 dashgo 和建图相关节点:

建图之前尽量保证空间封闭, 减小建立的地图尺寸。

roslaunch navigation\_task gmapping\_start.sh

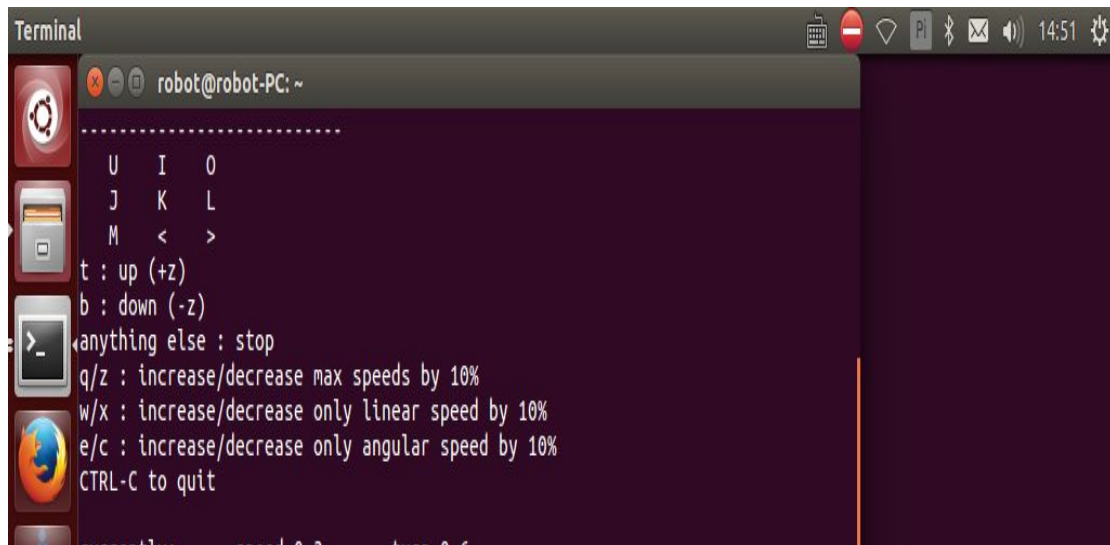
正常启动结果应于下图显示类似:

```
/home/robot/catkin_ws/src/dashgo/navigation_task/launch/gmapping_start.launch http://localhost:1
- linearUpdate 0.05 - angularUpdate 0.0436 - resampleThreshold 0.5
- xmin -5 - xmax 5 - ymin -6 - ymax 4 - delta 0.01 - particles 80
[ INFO] [1490079139.169449974]: Initialization complete
update frame 0
update ld=0 ad=0
Laser Pose= 0.316863 0 -1.52127
m_count 0
Registering First Scan
[ INFO] [1490079139.218807583]: Resizing costmap to 992 X 992 at 0.010000 m/pix
[ INFO] [1490079139.318612019]: Received a 992 X 992 map at 0.010000 m/pix
[ INFO] [1490079139.322443676]: Using plugin "obstacle_layer"
[ INFO] [1490079139.325229250]: Subscribed to Topics: laser_scan_sensor
[ INFO] [1490079139.345384408]: Using plugin "inflation_layer"
[ INFO] [1490079139.430347897]: Using plugin "static_layer"
[ INFO] [1490079139.454973443]: Requesting the map...
[ INFO] [1490079139.457684438]: Resizing static layer to 992 X 992 at 0.010000 m
/pix
[ INFO] [1490079139.557374621]: Received a 992 X 992 map at 0.010000 m/pix
[ INFO] [1490079139.560571810]: Using plugin "obstacle_layer"
[ INFO] [1490079139.562367953]: Subscribed to Topics: laser_scan_sensor
[ INFO] [1490079139.581914435]: Using plugin "inflation_layer"
[ INFO] [1490079139.639540822]: Created local_planner teb_local_planner/TebLocal
PlannerROS
[ WARN] [1490079139.695573921]: TebLocalPlannerROS() Param Warning: 'alternative
_time_cost' is deprecated. It has been replaced by 'selection_alternative_time_c
ost'.
[ INFO] [1490079139.700361152]: No robot footprint model specified for trajector
y optimization. Using point-shaped model.
[ INFO] [1490079139.700585036]: Parallel planning in distinctive topologies disa
bled.
[ INFO] [1490079139.700600267]: No costmap conversion plugin specified. All occu
pied costmap cells are treated as point obstacles.
[ INFO] [1490079140.413286340]: Recovery behavior will clear layer obstacles
[ INFO] [1490079140.435650147]: Recovery behavior will clear layer obstacles
[ INFO] [1490079140.479645320]: odom received!
update frame 37
update ld=0.0507964 ad=0
Laser Pose= 0.367659 0 -1.52127
m_count 1
Average Scan Matching Score=106.633
neff= 80
```

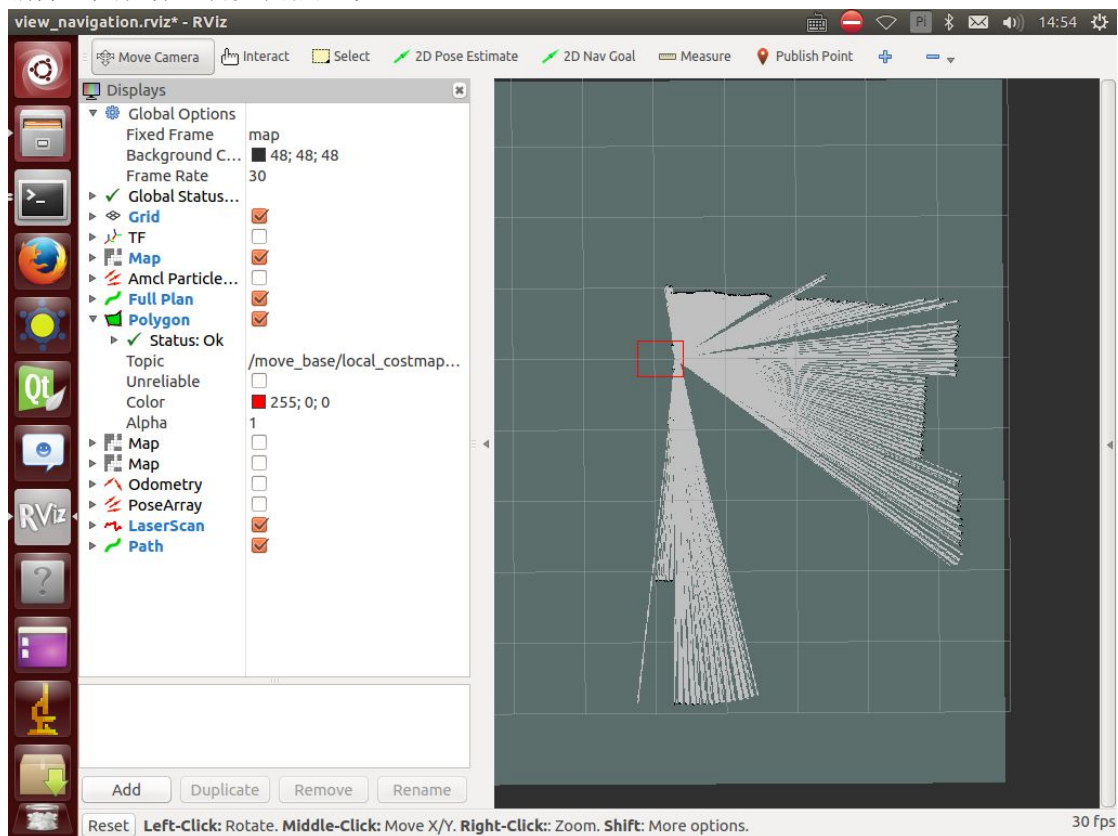
(3) 运行速度控制节点, 控制 dashgo 移动建立地图, 速度不要太快。

roslaunch dashgo\_tools teleop\_twist\_keyboard.py

线速度在 0.10m/s 左右 角速度 0.20rad/s



在打开的 rviz 中显示建立的地图，注意初始位置可能会有杂乱的点，重新扫描该位置消除。关键标志物要扫描足够。



(4) 通过 rviz 的显示，确保所需地图建立后，保存地图。

```
roslaunch navigation_task gmapping-save.sh
```

地图保存在 dashgo\_nav/maps 文件夹下，会覆盖掉之前的地图，注意做好备份。

## 2.5 导航

### 2.5.1 获取位置信息

(1) 对于新建立的地图，获取地图上关键目标点的位置：抓取位置和释放位置。

首先按照第一部分做好检查。运行导航相关节点

```
roslaunch navigation_task navigation-start.sh
```

```
Terminal
/home/robot/catkin_ws/src/dashgo/navigation_task/launch/navigation_imu.launch http://
robot@robot-PC:~$ rosruncatkin_ws/src/dashgo/navigation_task/launch/navigation_imu.launch http://
... logging to /home/robot/.ros/log/892408a0-0e03-11e7-ad82-91e0574b10be/roslaunch-robot-PC-3633.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://robot-PC:39698/

SUMMARY
-----
```

正常启动结果应于下图显示类似：

```
Terminal
/home/robot/catkin_ws/src/dashgo/navigation_task/launch/navigation_imu.launch http://
[ INFO ] [1490079405.359687360]: Using plugin "inflation_layer"
[ INFO ] [1490079405.433944564]: Using plugin "static_layer"
[ INFO ] [1490079405.457735843]: Requesting the map...
[ INFO ] [1490079405.460487840]: Resizing static layer to 992 X 992 at 0.010000 m/pix
[ INFO ] [1490079405.560214950]: Received a 992 X 992 map at 0.010000 m/pix
[ INFO ] [1490079405.564215628]: Using plugin "obstacle_layer"
[ INFO ] [1490079405.566163422]: Subscribed to Topics: laser_scan_sensor
[ INFO ] [1490079405.582387383]: Using plugin "inflation_layer"
[ INFO ] [1490079405.635704525]: Created local_planner teb_local_planner/TebLocalPlannerROS
[ WARN ] [1490079405.690955277]: TebLocalPlannerROS() Param Warning: 'alternative_time_cost' is deprecated. It has been replaced by 'selection_alternative_time_cost'.
[ INFO ] [1490079405.694290358]: No robot footprint model specified for trajectory optimization. Using point-shaped model.
[ INFO ] [1490079405.694545590]: Parallel planning in distinctive topologies disabled.
[ INFO ] [1490079405.694563871]: No costmap conversion plugin specified. All occupied costmap cells are treated as point obstacles.
[ INFO ] [1490079405.821982457]: Recovery behavior will clear layer obstacles
[ INFO ] [1490079405.845167823]: Recovery behavior will clear layer obstacles
[ INFO ] [1490079405.951827275]: odom received!
```

启动后，将 dashgo 电源关闭。

(2) rviz 中加载环境地图

roslaunch dashgo\_rviz view\_navigation.launch

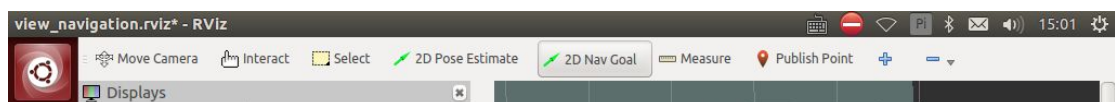
```
robot@robot-PC: ~
robot@robot-PC:~$ roslaunch dashgo_rviz view_navigation.launch
```

(3) 运行节点，获取位置信息

roslaunch navigation\_task get\_navigation\_position

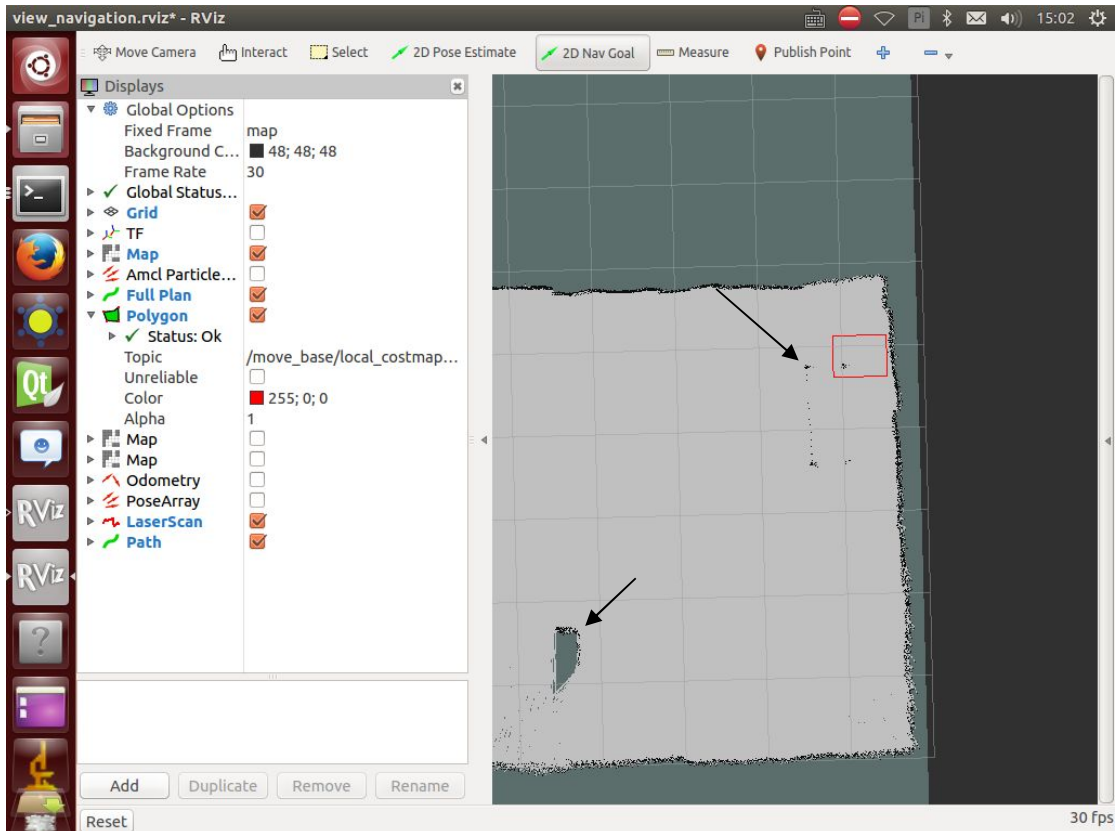
```
robot@robot-PC: ~
robot@robot-PC:~$ roslaunch navigation_task get_navigation_position
```

(4) 首先点击 2d nav goal





然后在地图中点击前方左侧立柱的位置，尽量靠外侧，接着点击放置位置的左上角，顺序不要反。



(5) 将 `get_navigation_position` 输出的坐标位置输入到 `home` 目录下的 `position.txt` 中。  
`pick1`、`pick2`、`pick3` 的 `x,y` 分别对应第 2,3,4 行的 1, 2 列，`place` 的 `x,y` 对应 5 行的 1, 2 列。

```
robot@robot-PC: ~
robot@robot-PC:~$ rosrunc navigation_task get_navigation_position
[ INFO] [1490079758.211775319]: Pick 1 X: 2.540,Y: -0.426
[ INFO] [1490079758.21184399]: Pick 2 X: 2.540,Y: -0.813
[ INFO] [1490079758.211912665]: Pick 3 X: 2.540,Y: -1.200
[ INFO] [1490079763.111766241]: Place 1 X: 1.175, Y: -2.675
robot@robot-PC:~$
```

该系列值需要通过测试微调。

(6) `position.txt` 文件说明：每一行对应一个目标点的位置和姿态。

第 1 列和第 2 列分别为 `x` 和 `y` 位置，第 3 列和第 4 列为姿态数据，分别为四元数中的 `Z` 和 `W`（2D 下，`X`，`Y` 为 0 省略）

测试时，建立地图的位置相对于场地原点的坐标（49.5 36），供参考

第 1 行 回原点位置：

位置	0.12	-0.15
姿态	0.707	0.707 (90 度)

第 2, 3, 4 行 抓取位置：

位置计算：`x` 部分均为：`position_x - 100` (70+30)  
`y` 部分分别为：`position_y - 0.234` (0.347/2+0.04 +0.02)  
`position_y - 0.621` (0.347/2+0.08 + 0.347 + 0.02)  
`position_y - 1.008` (0.347/2-0.12-0.347\*2 - 0.02)  
(`position_x`，`position_y`)为前方左侧立柱的位置，0.02 为调整值  
姿态部分均为 0 1 (0 度)

第 5 行 释放位置：

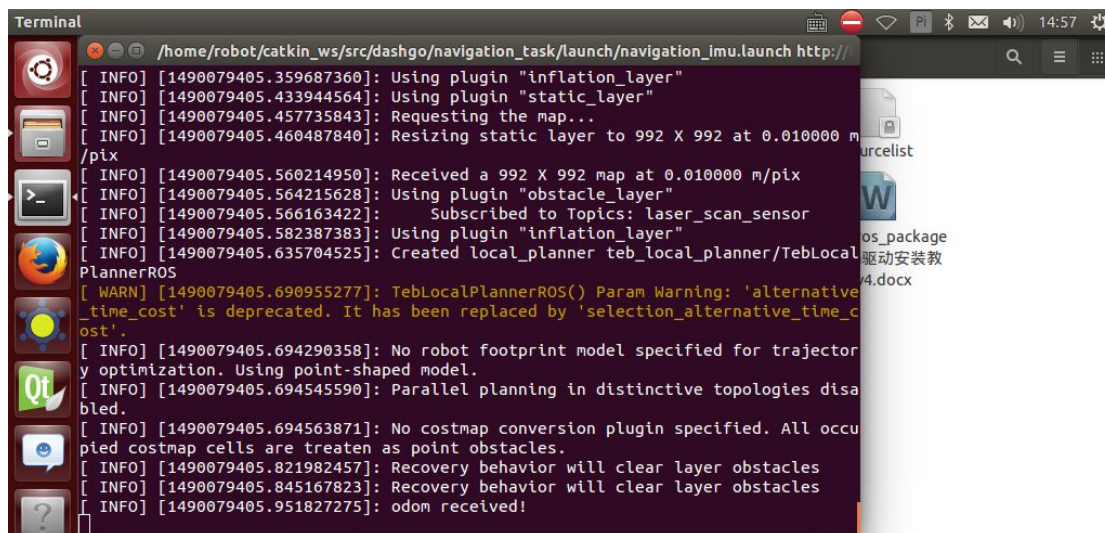
位置计算: x 部分为:  $\text{position\_x} + 0.46$  (0.10+0.24+0.12)  
y 部分为:  $\text{position\_y} + 0.65$  (0.10+0.33+0.21)  
(position\_x, position\_y)为放置区的左上角位置  
姿态: 0.707 -0.707

## 2.5.2 角度标定（第一次）

(1) 将移动平台与柜子对正，运行导航节点。

`roslaunch navigation_task navigation-start.sh`

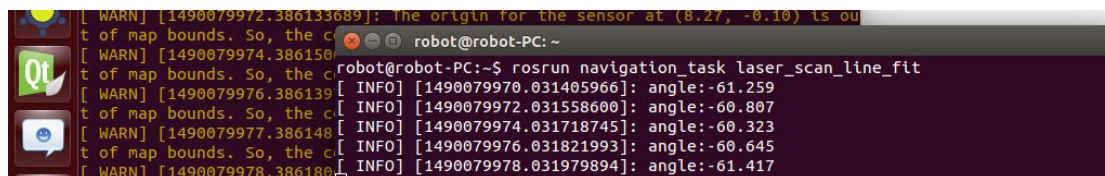
正常启动结果应于下图显示类似：



```
Terminal
/home/robot/catkin_ws/src/dashgo/navigation_task/launch/navigation_imu.launch http://
[ INFO] [1490079405.359687360]: Using plugin "inflation_layer"
[ INFO] [1490079405.433944564]: Using plugin "static_layer"
[ INFO] [1490079405.457735843]: Requesting the map...
[ INFO] [1490079405.460487840]: Resizing static layer to 992 X 992 at 0.010000 m
/pix
[ INFO] [1490079405.560214950]: Received a 992 X 992 map at 0.010000 m/pix
[ INFO] [1490079405.564215628]: Using plugin "obstacle_layer"
[ INFO] [1490079405.566163422]: Subscribed to Topics: laser_scan_sensor
[ INFO] [1490079405.582387383]: Using plugin "inflation_layer"
[ INFO] [1490079405.635704525]: Created local_planner teb_local_planner/TebLocal
PlannerROS
[ WARN] [1490079405.690955277]: TebLocalPlannerROS() Param Warning: 'alternative
_time_cost' is deprecated. It has been replaced by 'selection_alternative_time_c
ost'.
[ INFO] [1490079405.694290358]: No robot footprint model specified for trajectory
optimization. Using point-shaped model.
[ INFO] [1490079405.694545590]: Parallel planning in distinctive topologies disa
bled.
[ INFO] [1490079405.694563871]: No costmap conversion plugin specified. All occu
pied costmap cells are treated as point obstacles.
[ INFO] [1490079405.821982457]: Recovery behavior will clear layer obstacles
[ INFO] [1490079405.845167823]: Recovery behavior will clear layer obstacles
[ INFO] [1490079405.951827275]: odom received!
```

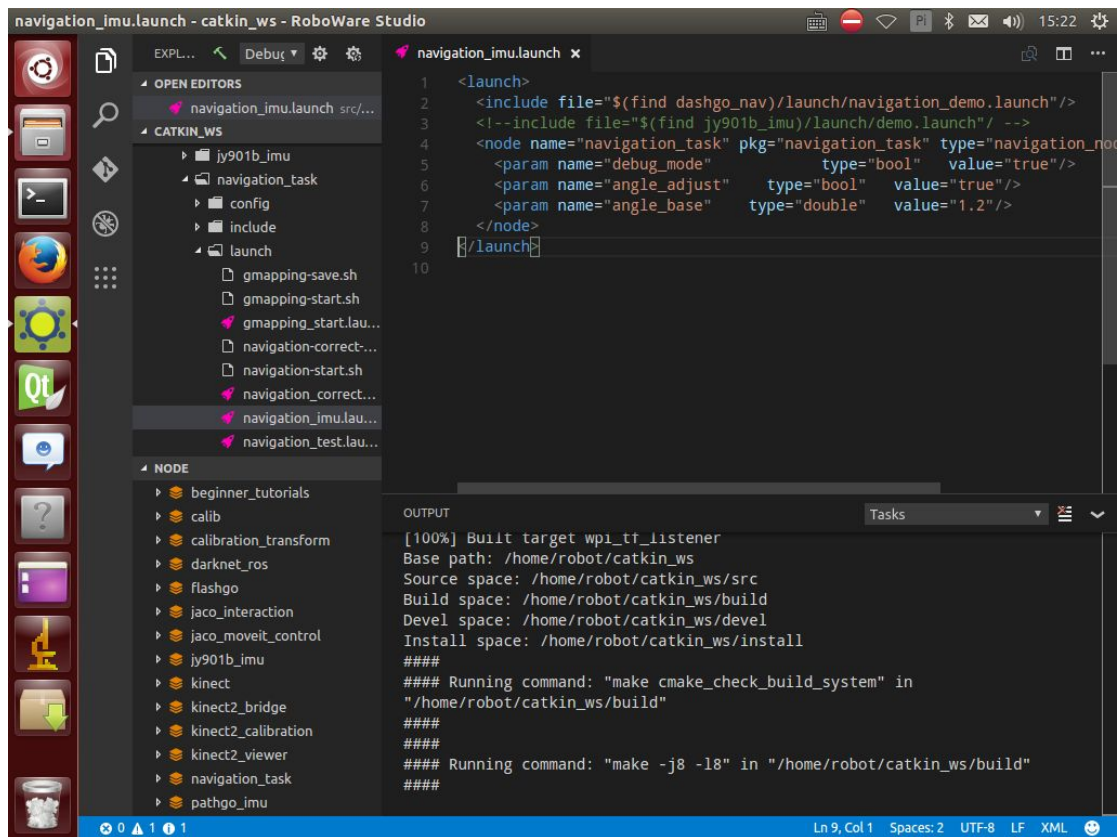
(2) 运行角度获取节点。

`roslaunch navigation_task laser_scan_line_fit`



```
robot@robot-PC: ~
robot@robot-PC:~$ roslaunch navigation_task laser_scan_line_fit
[ WARN] [1490079972.386133689]: The origin for the sensor at (8.27, -0.10) is ou
t of map bounds. So, the c
[ WARN] [1490079974.386150
t of map bounds. So, the c
[ WARN] [1490079976.386139
t of map bounds. So, the c
[ WARN] [1490079977.386148
t of map bounds. So, the c
[ INFO] [1490079970.031405966]: angle:-61.259
[ INFO] [1490079972.031558600]: angle:-60.807
[ INFO] [1490079974.031718745]: angle:-60.323
[ INFO] [1490079976.031821993]: angle:-60.645
[ INFO] [1490079978.031979894]: angle:-61.417
```

读取角度，将该值的相反数写入到 navigation\_task 包下的 navigation\_imu.launch 中的 angle\_base 项。



调试过程中，若固定偏向某一个方向，右偏给负值补偿，左偏给正值补偿。

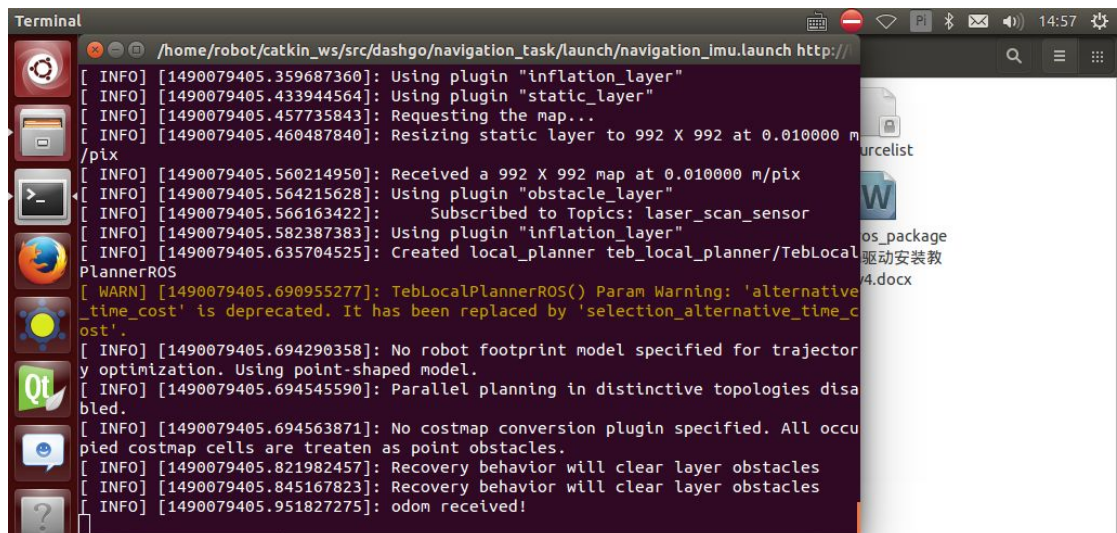
### 2.5.3 导航

#### (1) 运行导航节点

将移动平台放置在做好标记的初始摆放位置，摆正，按照第一部分做好检查，关闭其他节点，运行。

`roslaunch navigation_task navigation-start.sh`

正常启动结果应于下图显示类似：



#### (2) 运行流程控制节点。

`roslaunch beginner_tutorials ge_test`