Turtlebot3 SLAM Project

1. Introduction to the Project

TurtleBot3 supports development environment that can be programmed and developed with a virtual robot in the simulation. There are two development environments to do this, one is using fake node and 3D visualization tool RViz and the other is using the 3D robot simulator Gazebo.

The fake node method is suitable for testing with the robot model and movement, but it cannot use sensors. If you need to test SLAM and Navigation, we recommend using Gazebo, which can use sensors such as IMU, LDS, and camera in the simulation.

2. Software Installation

2.1 Install Dependent Packages

First of all you need to install the dependent packages using the following command.

```
sudo apt-get install ros-melodic-joy ros-melodic-teleop-twist-joy
ros-melodic-teleop-twist-keyboard ros-melodic-laser-proc ros-melodic-
rgbd-launch ros-melodic-depthimage-to-laserscan ros-melodic-
rosserial-arduino ros-melodic-rosserial-python ros-melodic-rosserial-
server ros-melodic-rosserial-client ros-melodic-rosserial-msgs ros-
melodic-amcl ros-melodic-map-server ros-melodic-move-base ros-
melodic-urdf ros-melodic-xacro ros-melodic-compressed-image-transport
ros-melodic-rqt-image-view ros-melodic-gmapping ros-melodic-
navigation ros-melodic-interactive-markers
```

2.2 Install Turtlebot3 Packages

Then, you can download the Turtlebot3 stack and packages using the following commands, pls run the commands line by line in your Ubuntu shell.

```
mkdir -p ~/project1_ws/src
cd ~/project1_ws/src
catkin_init_workspace
git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
git clone  https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
cd ~/catkin_ws && catkin_make
echo "source ~/project1_ws/devel/setup.bash" >> ~/.bashrc
```

In Ubuntu Linux, the hidden file in your home directory, .bashrc, is automatically run everytime you open a new shell. It is very important in ROS that you need to include your newly compiled packages in the environment variable ROS_PACKAGE_PATH so that ros master can recognize and know where to find it. The way to do it is to source the setup script file, like run the command "*source ~/catkin_ws/devel/setup.bash*" in your shell. DO NOT FORGET IT! Otherwise you cannot run your ROS packages.A convenient way is to put this command in the .bashrc so that it will be automatically executed whenever you open a new shell.

3. Turtlebot3 Simulation

3.1 Turtlebot3 Simulation using Fake Node

https://youtu.be/iHXZSLBJHMg

To launch the virtual robot, execute the `turtlebot3_fake.launch` file in the `turtlebot3_fake` package as shown below. The `turtlebot3_fake` is a very simple simulation node that can be run without having an actual robot. You can even control the virtual TurtleBot3 in RViz with a teleoperation node.

Before executing this command, you have to specify the model name of TurtleBot3. The `${TB3_MODEL}` is the name of the model you are using in `burger`, `waffle`, `waffle_pi`.

So everytime before you run a ROS node, you need to run export TURTLEBOT3_MODEL=${TB3_MODEL}, where ${TB3_MODEL} is burger, waffle or waffle_pi. To permanently set this environment variable, you can attach the following line in your .bashrc hidden file.

```
export TURTLEBOT3_MODEL=burger
```

Let's run the following commands to do the simulation. In a shell, run the following:

```
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_fake turtlebot3_fake.launch
```

Open a new shell, run the following to teleoperate the robot:

```
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

3.2 Turtlebot3 Simulation using Gazebo

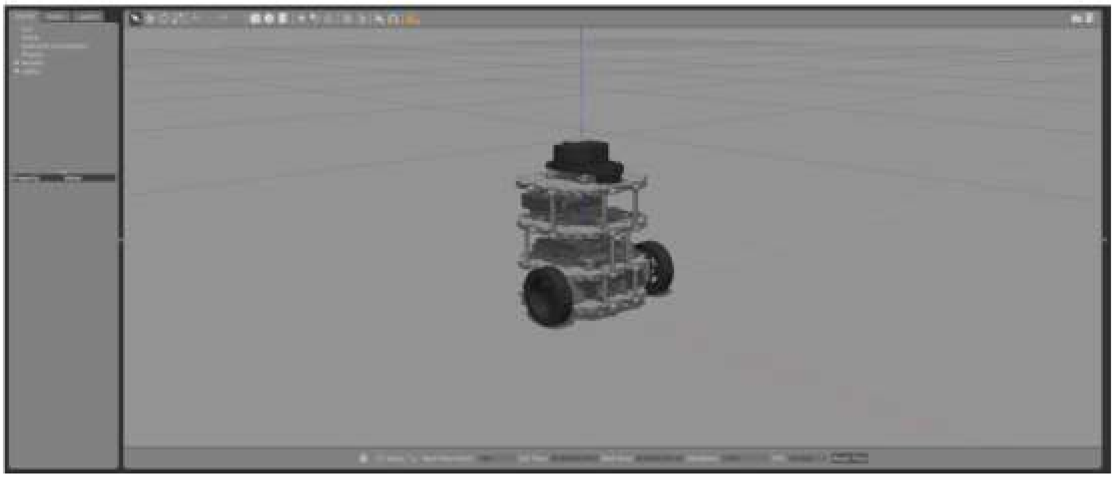
https://youtu.be/Uz0oJ6a_m0g

There are two ways to simulate using Gazebo. first method is to use with ROS through turtlebot3_gazebo package and second method is to use only gazebo and turtlebot3_gazebo_plugin plugin without using ROS. We will focus on the first way.

3.2.1 Simulation in Empty World

The following command can be used to test the virtual TurtleBot3 on the empty world of the gazebo default environment.

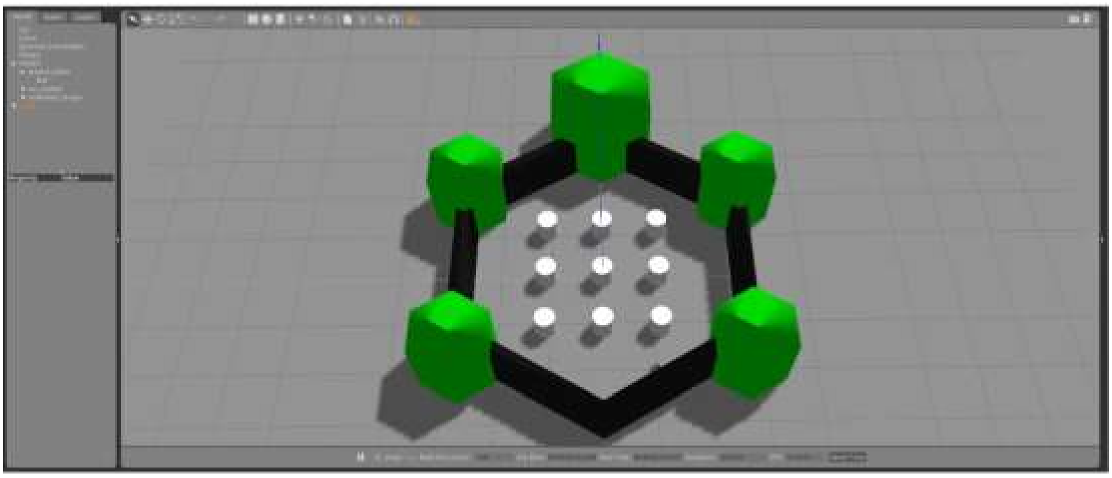
```
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
```



3.2.2 Simulation in Turtlebot3 World

TurtleBot3 world is a map consists of simple objects that makes up the shape of TurtleBot3 symbol. TurtleBot3 world is mainly used for testing such as SLAM and Navigation.

```
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

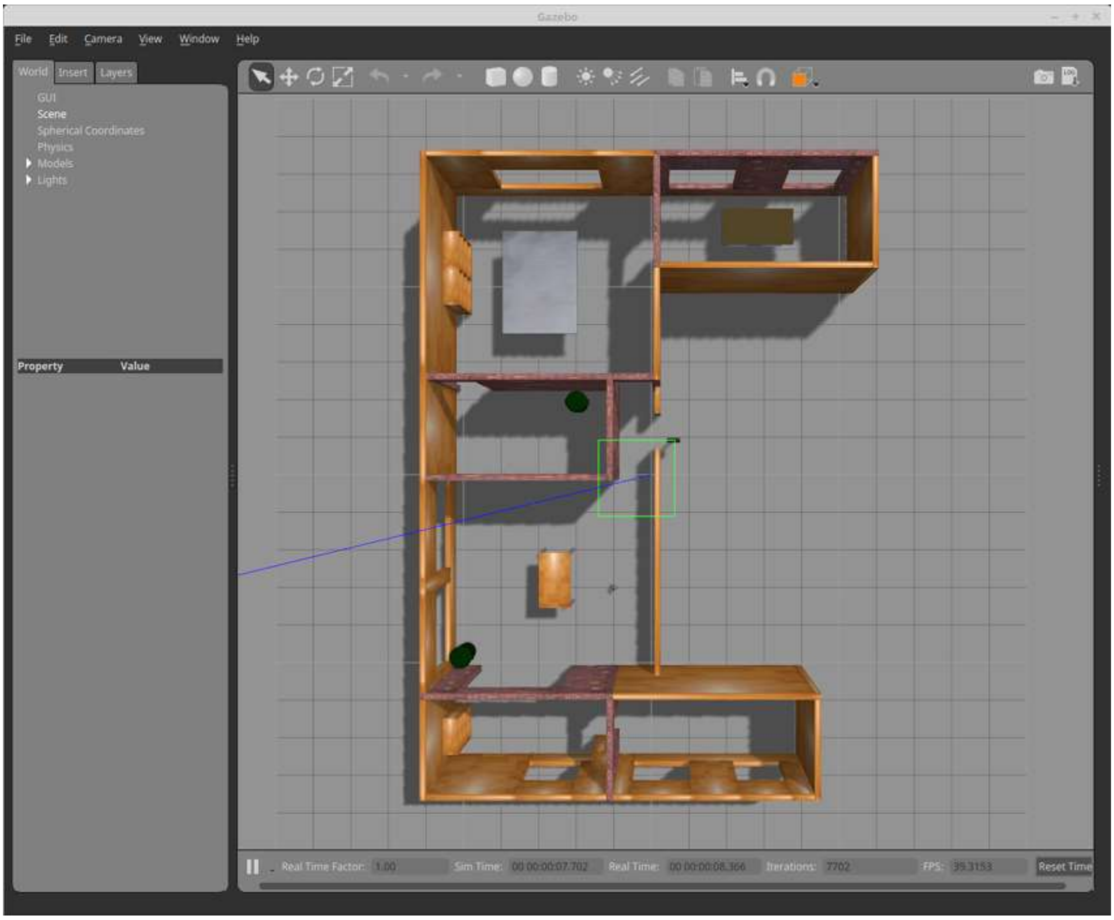


3.2.3 Simulation in Turtlebot3 House

TurtleBot3 House is a map made with house drawings. It is suitable for testing related to more complex task performance.

```
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_gazebo turtlebot3_house.launch
```

NOTE : If TurtleBot3 House is excuted for the first time, downloading the map file takes a couple of minutes or more depending on download speed. You need internet connection.



3.3 Drive Turtlebot3

3.3.1 Teleoperation on Gazebo

In order to control a TurtleBot3 with a keyboard, please launch teleoperation feature with below command in a new terminal window.

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

3.3.2 Collision Avoidance

In order to autonomously drive a TurtleBot3 around the **TurtleBot3 world**, open a new terminal window and enter below command.

```
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

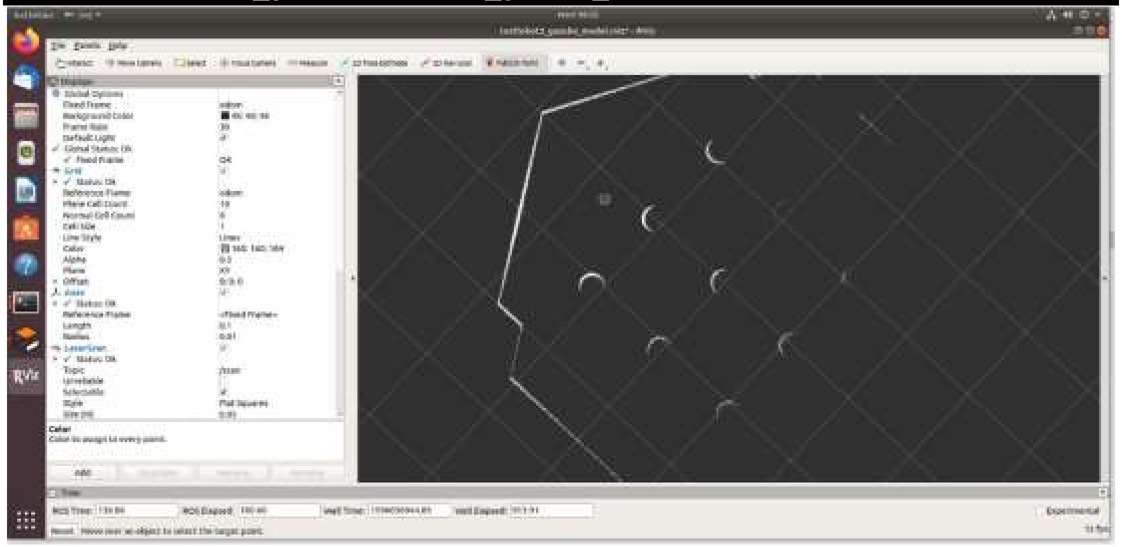
Open a new terminal window and enter below command.

```
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_gazebo turtlebot3_simulation.launch
```

3.4 Execute rviz

RViz visualizes published topics while simulation is running. You can launch RViz in a new terminal window by entering below command.

```
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch
```



3.5 Virtual SLAM with Turtlebot3

For virtual SLAM in Gazebo, instead of running the actual robot, you can select the various environments and robot models mentioned above, and the SLAM-related commands will use the ROS packages used in the SLAM section.

3.5.1 Virtual SLAM Execution Procedure

The following commands are examples of using the TurtleBot3 Waffle Pi model and the turtlebot3_world environment.

- Launch Gazebo

```
export TURTLEBOT3_MODEL=waffle_pi
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

- Launch SLAM

```
export TURTLEBOT3_MODEL=waffle_pi
roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
```

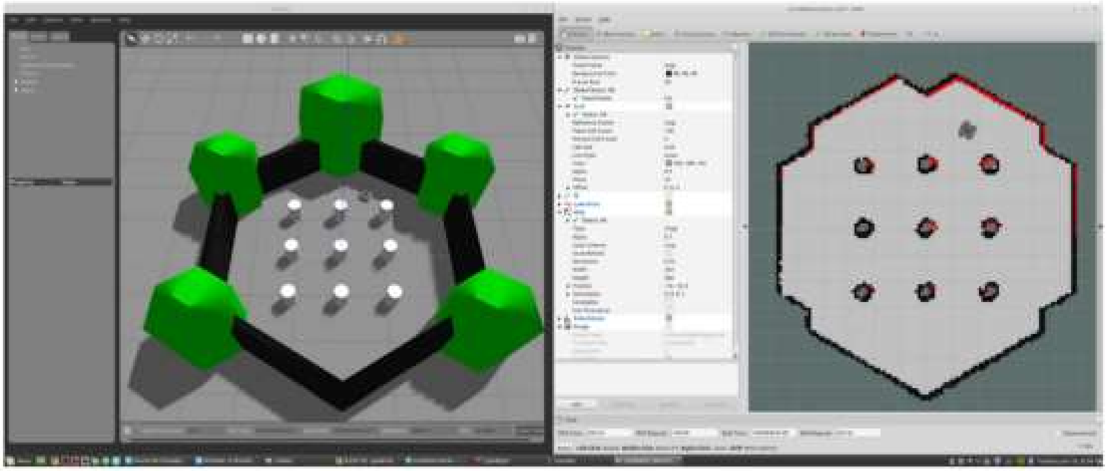
- Remotely Control TurtleBot3

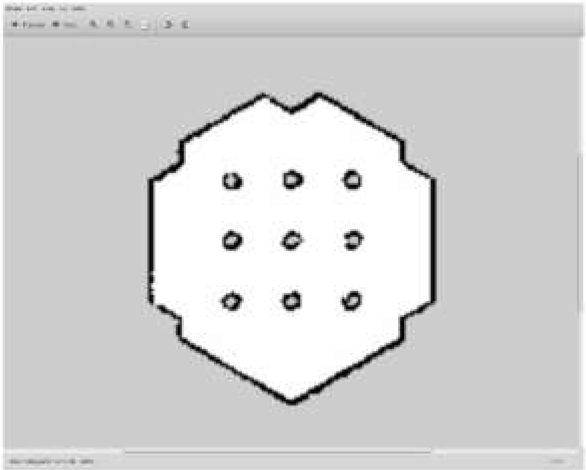
```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

- Save the Map

```
rosrun map_server map_saver -f ~/map
```

When you run the dependent packages and move the robot in virtual space and create a map as shown above, you can create a map as shown in figure below.





3.6 Virtual Navigation with Turtlebot3

For virtual Navigation in Gazebo, instead of running the actual robot, you can select the various environments and robot models mentioned above, and the Navigation-related commands will use the ROS packages used in the Navigation section.

3.6.1 Virtual Navigation Execution Procedure

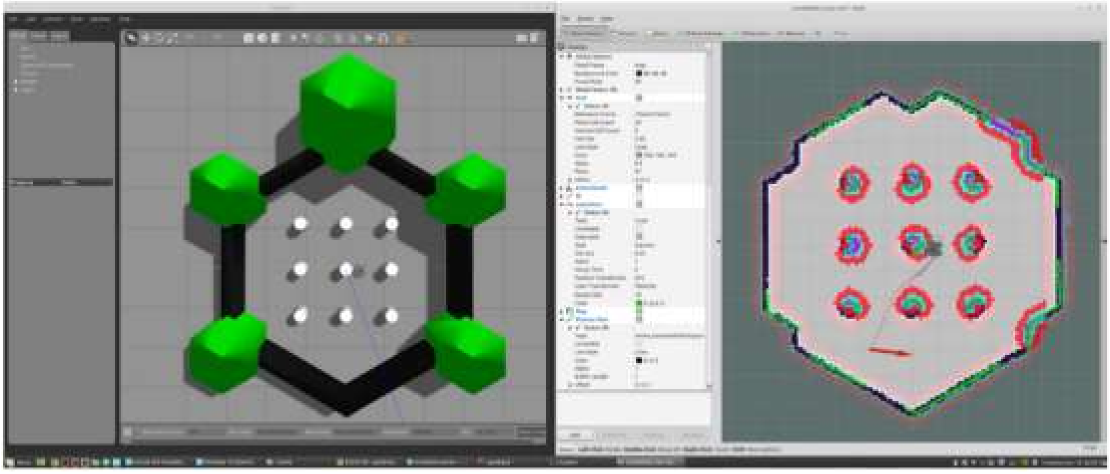
Terminate all applications that were executed during the virtual SLAM practice and execute related packages in the following instruction, the robot will appear on the previously generated map. After setting the initial position of the robot on the map, set the destination to run the navigation as shown in figure below. The initial position only needs to be set once.

- Execute Gazebo

```
export TURTLEBOT3_MODEL=waffle_pi
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

- Execute Navigation

```
export TURTLEBOT3_MODEL=waffle_pi
roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```



3.7 Testing Virtual Turtlebot3 by Tuning Parameters

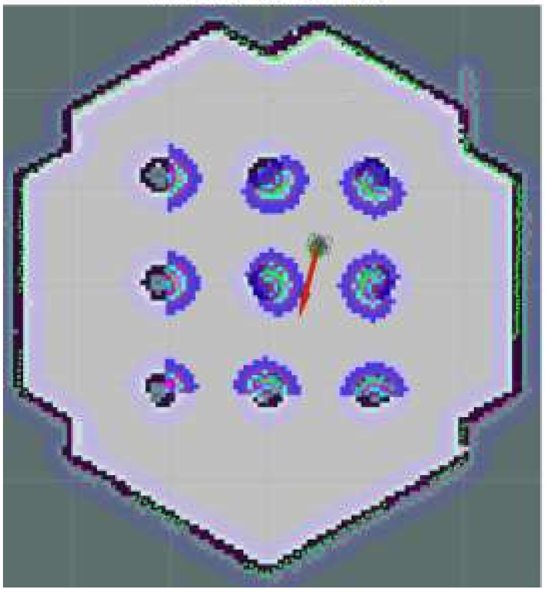
Navigation stack has many parameters to change performances for different robots. You can get an information about it in [ROS Wiki](). This tuning guide give some tips for you to configue important parameters. If you want to change performances depends on your environments, this tips might be help you and save your time.

3.7.1 inflation_radius

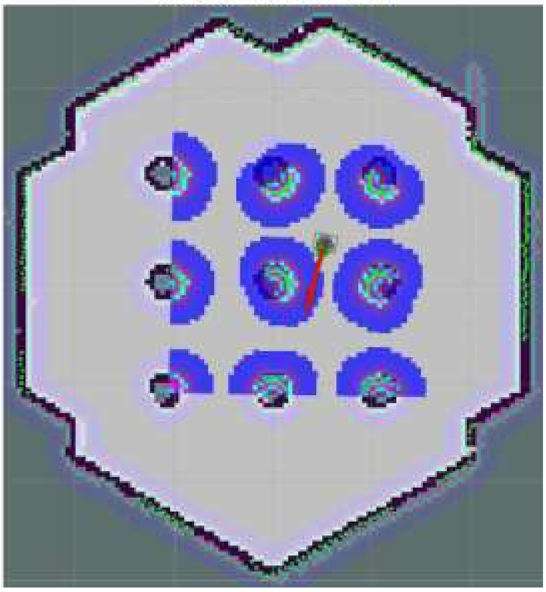
```
turtlebot3_navigation/param/costmap_common_param_$(model).yaml
```

This parameter makes inflation area from the obstacle. Path would be planned in order that it doesn't cross this area. It is safe that to set this to be bigger than robot radius.

inflation radius = 0.2



inflation radius = 1.0



3.7.2 cost_scaling_factor

```
turtlebot3_navigation/param/costmap_common_param_$(model).yaml
```

This factor is multiplied by cost value. Because it is a reciprocal proportion, this parameter is increased, the cost is decreased. The best path is for the robot to pass through a center of between obstacles. Set this factor to be smaller in order to far from obstacles.

3.7.3 Other Parameters

max_vel_x

turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml

This factor is set the maximum value of translational velocity.

min_vel_x

turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml

This factor is set the minimum value of translational velocity. If set this negative, the robot can move backwards.

max_trans_vel

turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml

Actual value of the maximum translational velocity. The robot can not be faster than this.

min_trans_vel

turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml

Actual value of the minimum translational velocity. The robot can not be slower than this.

max_rot_vel

turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml

Actual value of the maximum rotational velocity. The robot can not be faster than this.

min_rot_vel

turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml

Actual value of the minimum rotational velocity. The robot can not be slower than this.

acc_lim_x

turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml

Actual value of the translational acceleration limit.

acc_lim_theta

turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml

Actual value of the rotational acceleration limit.

xy_goal_tolerance

turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml

The x,y distance allowed when the robot reaches its goal pose.

yaw_goal_tolerance

turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml

The yaw angle allowed when the robot reaches its goal pose.

Exercise: Please change the inflation_radius to see the navigation effect.

4. Conclusion

SLAM and navigation are two important areas in intelligent robotics. ROS has realized the SLAM algorithm, namely gmapping, and the navigation algorithm, namely acml. In this project, we have done a range of exercises to study how to use the related ROS packages to do SLAM and navigation for the Robotis Turtlebot3. It is expected that you will be able to know the basic knowledge of the ROS SLAM and navigation applications so that you will be able to do the similar exercises using other third party packages.

Reference:

1. https://emanual.robotis.com/docs/en/platform/turtlebot3