ELEG4701: Intelligent Interactive Robot Practice

Lecture 6: ROS Navigation

Jiewen Lai

Research Assistant Professor

jwlai@ee.cuhk.edu.hk

EE, CUHK

2024 Spring

Outline

Part 1. Preliminaries

Preliminaries

Following the instructions in the previous lectures, you should have:

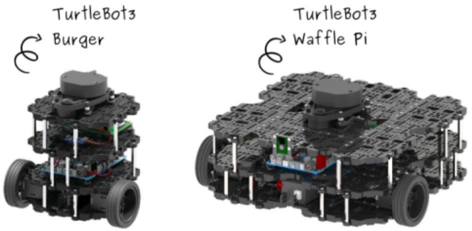
- Installed Ubuntu 20.04 in Virtual Machine (VM)
- Installed ROS (noetic) on Ubuntu
- Basic understanding of ROS
- Understood concepts like nodes, packages (camkelists.txt; package.xml), CMake process, dependencies, publisher/subscriber, topic & msg, service/client, srv, launch & tag
- Utilized these concepts to control 🐢🐢🐢🐢🐢

For the next stage of this course, you will:

- **Explore the TurtleBot3 simulation world in ROS**
- Learn to write a simple ROS node to listen to the simulated Lidar sensor

Preliminaries

TURTLEBOT3



Ref: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

Preliminaries

1. Play with simulators in ROS – ***Gazebo***
2. Understand the workflow of the ROS navigation stack
3. Build a map in simulation
4. Navigate with a known map in the simulation
5. Use the RViz tool for visualization

Preliminaries

About Debugging

If anything goes wrong, try to debug using the command-line tools in ROS such as `rostopic` and `rosnode`.

Some commonly used commands for debugging:

To list active topics

```
$ rostopic list
```

To print the info of the topic

```
$ rostopic info /topic_name
```

Preliminaries

To print messages of the topic

```
$ rostopic echo /topic_name
```

To list active nodes

```
$ rosnode list
```

To print info about the node

```
$ rosnode info /node_name
```

Please refer to http://wiki.ros.org/ROS/CommandLineTools for a detailed tutorial.

Some suggestions:

- Google the errors
- Google the errors
- Google the errors
- Make use of ROS wiki and ROS answers
- Read the documentation of the ROS packages we use
- https://stackoverflow.com/questions
- 中文 Web forums & blog for developers: CSDN, Zhihu, etc.

Part 2. Simulators (Task 1)

Simulators

Install TurtleBot (Plain text available at lab6-memo.txt)

Install Dependent ROS Packages

```
$ sudo apt install ros-noetic-gazebo-ros-pkgs
ros-noetic-gazebo-ros-control
$ sudo apt-get install ros-noetic-turtlebot-*
$ sudo apt install ros-noetic-gmapping
```

Install Dependent ROS Packages

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

Simulators

Establish WS -> Git Clone -> CMake

```
$ cd catkin_ws #(you can mkdir a new workspace too)
$ cd src #(you need to mkdir src if you're going to create a new ws)
$ 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_make
$ source ~/catkin_ws/devel/setup.bash
```

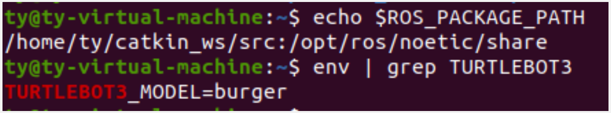
Simulators

Configure the environment

Add an environmental statement for your work environment

```
$ echo "export TURTLEBOT3_MODEL=burger" >> ~/.bashrc
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

If you saw this it means the env setting is valid



```
ty@ty-virtual-machine:~$ echo $ROS_PACKAGE_PATH
/home/ty/catkin_ws/src:/opt/ros/noetic/share
ty@ty-virtual-machine:~$ env | grep TURTLEBOT3
TURTLEBOT3_MODEL=burger
```

More about: apt vs. apt-get

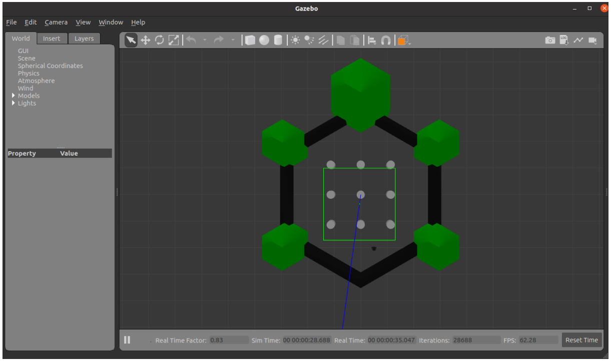
- Both **apt** and **apt-get** are command line tools
- Debian-based Linux operating systems use the Advanced Package Tool (APT) to manage Linux software packages
- The **apt** command is a more user-friendly package manager than **apt-get**
- More on: https://aws.amazon.com/compare/the-difference-between-apt-and-apt-get/

Simulators

3D simulator: Gazebo

Open a terminal in Ubuntu and launch the simulator:

```
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```



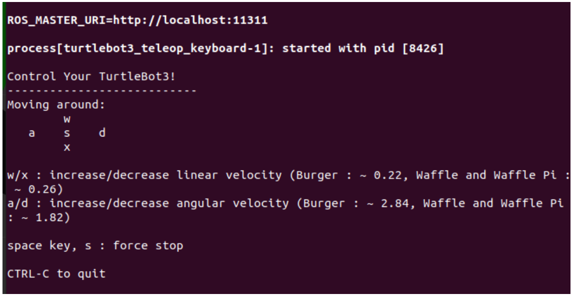
Real Time Factor: 0.83 Sim Time: 00 00:00:28.688 Real Time: 00 00:00:35.047 Iterations: 28688 FPS: 62.28

Simulators

Next, let's try to use the keyboard to teleop the robot in the simulator.

Open a **new** terminal then run:

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```



```
ROS_MASTER_URI=http://localhost:11311

process[turtlebot3_teleop_keyboard-1]: started with pid [8426]

Control Your TurtleBot3!
---------------------------
Moving around:
        w
   a    s    d
        x

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi :
 ~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi
: ~ 1.82)

space key, s : force stop

CTRL-C to quit
```

Part 3. An Overview of ROS Navigation

An Overview of ROS Navigation

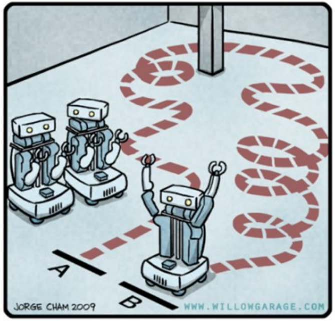
What is "navigation"?

Navigation is concerned with finding the way to a desired destination.

In order to reach a goal position, a mobile robot must:

- have a map of the environment,
- perceive its surroundings,
- localize itself, and
- plan its movement.

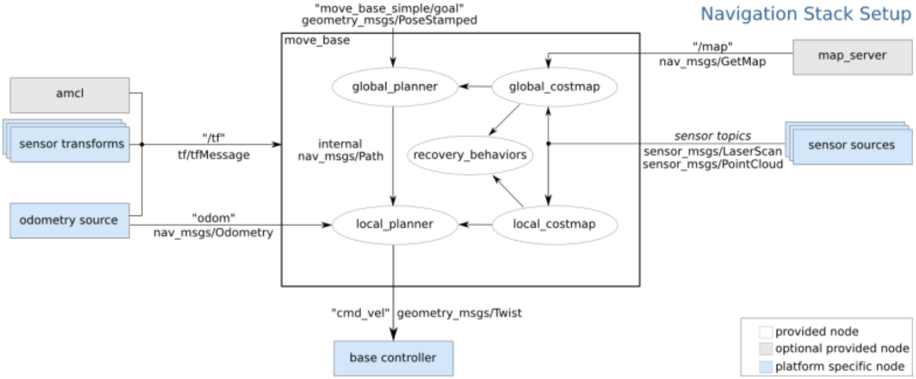
R.O.B.O.T. Comics



"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

An Overview of ROS Navigation

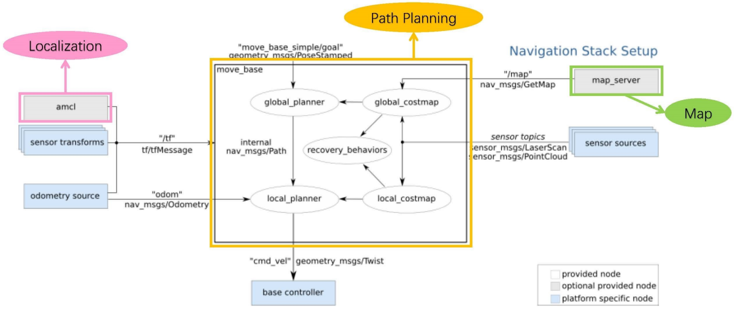
The ROS **Navigation Stack** serves to drive a mobile base to the goal position while safely avoiding obstacles.



An Overview of ROS Navigation

Important nodes:

- `gmapping` (before navigation): build a map
- `map_server`: offer the map data
- `amcl`: localize itself in the map (Adaptive Monte Carlo Localization)
- `move_base`: plan a path and drive to the goal



Part 4. Build a Map in Simulation (Task 2)

Build a Map in Simulation

How to build a map for navigation?

Simultaneous Localization and Mapping (SLAM) deals with creating a map of an unknown environment while simultaneously keeping track of an agent's location within it.

Refer to the book *Probabilistic Robotics*[1] for more details.

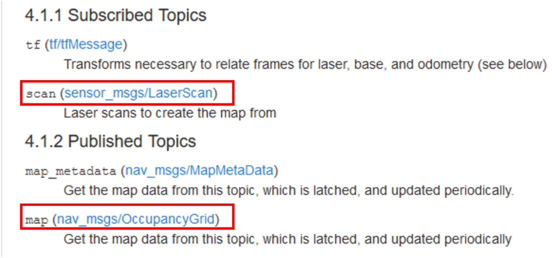
[1] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics. MIT Press. 2005

Build a Map in Simulation

gmapping – a laser-based SLAM algorithm

The gmapping package in ROS can create a 2D occupancy grid map from the laser and pose data collected by a mobile robot.

Take a look at the slam gmapping node in ROS (Publisher/Subscriber):



Takes in laser scans

Builds a map

4.1.1 Subscribed Topics

`tf` (tf/tfMessage)
Transforms necessary to relate frames for laser, base, and odometry (see below)

`scan` (sensor_msgs/LaserScan)
Laser scans to create the map from

4.1.2 Published Topics

`map_metadata` (nav_msgs/MapMetaData)
Get the map data from this topic, which is latched, and updated periodically.

`map` (nav_msgs/OccupancyGrid)
Get the map data from this topic, which is latched, and updated periodically

There are also many other ROS packages for other SLAM algorithms. For more information, please refer to http://wiki.ros.org/

Build a Map in Simulation

First, run Gazebo

Terminal 1

```
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

Open a new terminal and run gmapping

Terminal 2

```
$ roslaunch turtlebot3_slam turtlebot3_slam.launch
slam_methods:=gmapping
```

Now, the robot can “see” part of the surrounding environment where it is standing.

Build a Map in Simulation

We can use the teleop tool to make it wander around and build a map of the simulated environment.

Open a new terminal, run

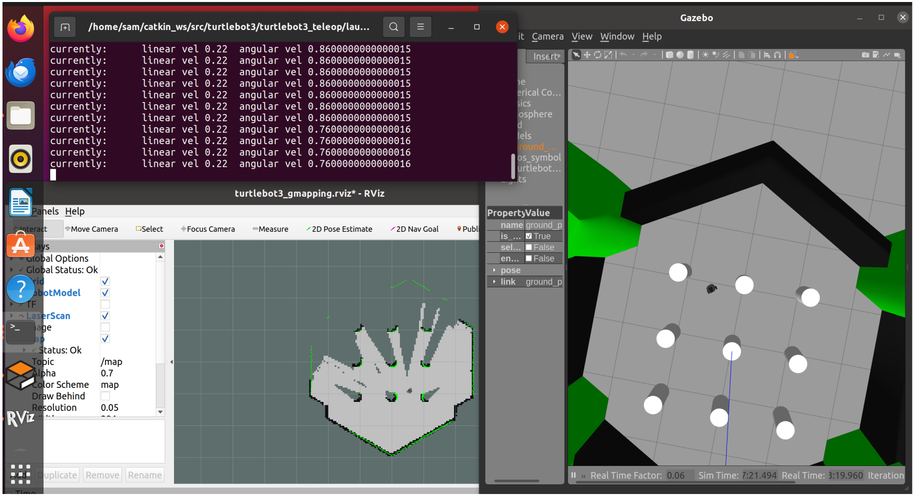
Terminal 3

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Use the keyboard to move the robot around carefully.

Build a Map in Simulation

Teleop to navigate:



```
/home/sam/catkin_ws/src/turtlebot3/turtlebot3_teleop/lau...
currently:      linear vel 0.22  angular vel 0.8600000000000015
currently:      linear vel 0.22  angular vel 0.8600000000000015
currently:      linear vel 0.22  angular vel 0.8600000000000015
currently:      linear vel 0.22  angular vel 0.8600000000000015
currently:      linear vel 0.22  angular vel 0.8600000000000015
currently:      linear vel 0.22  angular vel 0.8600000000000015
currently:      linear vel 0.22  angular vel 0.8600000000000015
currently:      linear vel 0.22  angular vel 0.7600000000000016
currently:      linear vel 0.22  angular vel 0.7600000000000016
currently:      linear vel 0.22  angular vel 0.7600000000000016
currently:      linear vel 0.22  angular vel 0.7600000000000016
```

Build a Map in Simulation

After the map is built, open a new terminal to save the map:

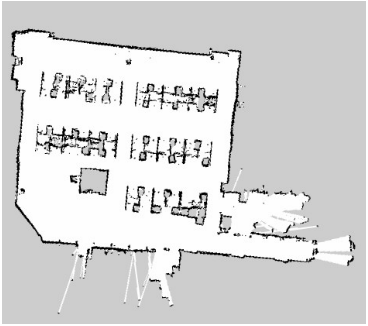
Terminal 4

```
$ rosrun map_server map_saver -f ~/my_map
```

Shut down the nodes after you save the map.

Build a Map in Simulation

Then, you should have successfully built a map. Below is a map of SHB210 (if you are using a physical TURTLEBOT3).

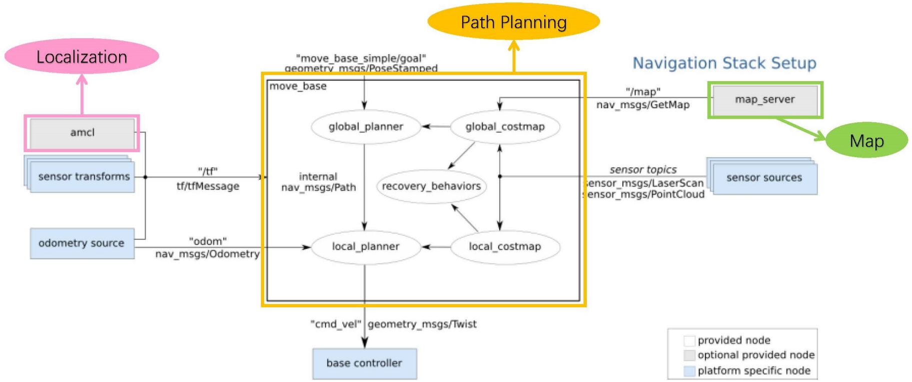


More on:
https://emanual.robotis.com/docs/en/platform/turtlebot3/nav_simulation/

Part 5. Navigation with a Known Map in Simulation (Task 3)

Navigate with a Known Map in Simulation

Remember the **three** important components of navigation:



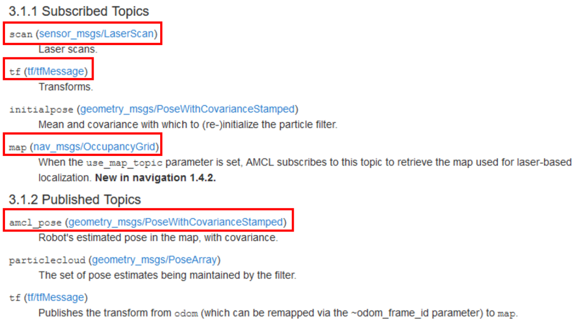
Navigate with a Known Map in Simulation

AMCL –a laser-based probabilistic localization system

The amcl package implements the **Adaptive Monte Carlo Localization** approach, which uses a **particle filter** to track the robot pose in a known map. Refer to the book *Probabilistic Robotics* for more details. Take a look at the amcl node in ROS:

Takes in laser scans, transforms, and a laser-based map

Outputs pose estimates



3.1.1 Subscribed Topics

`scan` (sensor_msgs/LaserScan)
Laser scans.

`tf` (tf/tfMessage)
Transforms.

`initialpose` (geometry_msgs/PoseWithCovarianceStamped)
Mean and covariance with which to (re-)initialize the particle filter.

`map` (nav_msgs/OccupancyGrid)
When the `use_map_topic` parameter is set, AMCL subscribes to this topic to retrieve the map used for laser-based localization. **New in navigation 1.4.2.**

3.1.2 Published Topics

`amcl_pose` (geometry_msgs/PoseWithCovarianceStamped)
Robot's estimated pose in the map, with covariance.

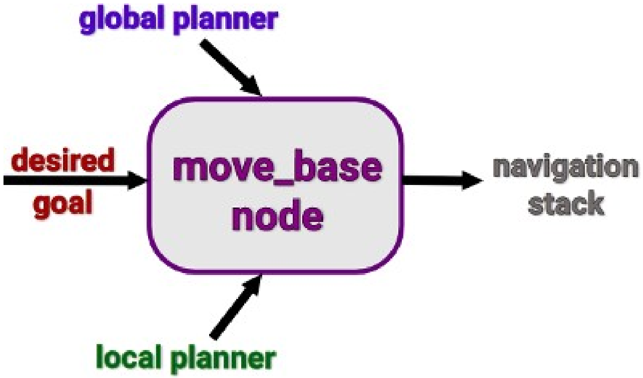
`particlecloud` (geometry_msgs/PoseArray)
The set of pose estimates being maintained by the filter.

`tf` (tf/tfMessage)
Publishes the transform from `odom` (which can be remapped via the ~odom_frame_id parameter) to `map`.

Navigate with a Known Map in Simulation

move_base –an implementation of actions to reach the goal

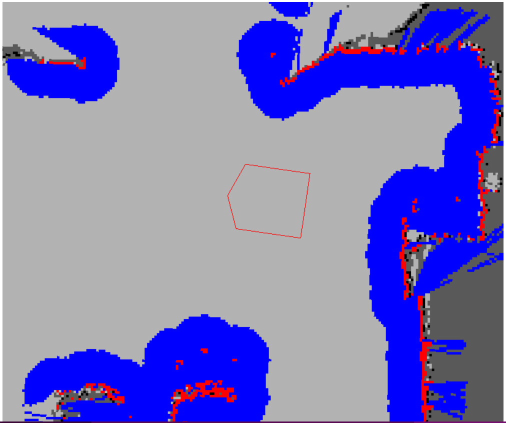
- Given a goal in the world, the move_base package can provide actions that attempt to reach it with a mobile base.
- The move_base node links together a global and local planner to accomplish its global navigation task.



Navigate with a Known Map in Simulation

costmap_2d –occupancy grid built from sensor data

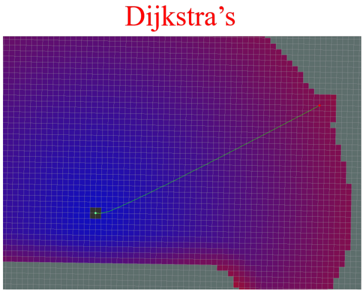
- The costmap_2d package can build a 2D occupancy grid of the sensor data, which tells the robot where it should navigate.
- The move base node links two costmaps, one for the global planner, one for the **global planner**, and one for the **local planner**.

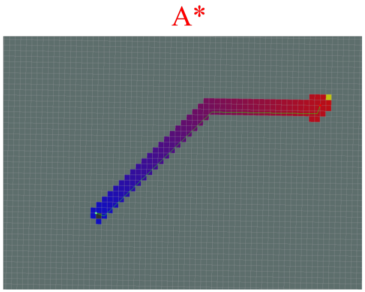


Navigate with a Known Map in Simulation

Global Planner –plan a trajectory

Given the (1) map, (2) the robot's current position, and (3) the goal position, the global planner produces a path between the two points.





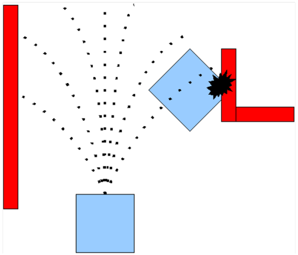
- Dijkstra, E. W. (1959). A Note on Two Problems in Connection with Graphs. *Numeriche Mathematik*, 1:269-271.
- Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100-107.

Navigate with a Known Map in Simulation

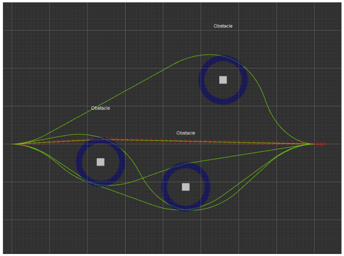
Local Planner –determine the current motion

Given a global plan to follow and a costmap, the local planner produces velocity commands to send to a mobile base.

Dynamic Window Approach (DWA)



Timed Elastic Band (TEB)

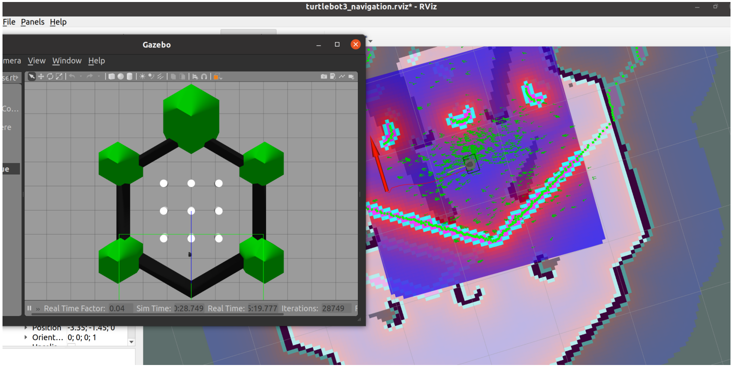


- Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1), 23-33.
- Rsmann, C., Hoffmann, F., & Bertram, T. (2017). Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems*, 88, 142-153.

Navigate with a Known Map in Simulation

Bring up Gazebo then run the navigation

```
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch
map_file:=$HOME/my_map.yaml
```


turtlebot3_navigation.rviz* - RViz
File Panels Help
Gazebo
mera View Window Help
Real Time Factor: 0.04 Sim Time:):28.749 Real Time: 5:19.777 Iterations: 28749
Position -3.35; -1.45; 0
Orient… 0; 0; 0; 1

Navigate with a Known Map in Simulation

1. **Use RViz to initialize the robot pose:**
   Click on the "2D Pose Estimate" button, and draw an estimated pose of the robot in the map. You will see the robot pose on the map is changed.
2. **Use RViz to send a goal to the robot:**
   Click on the "2D Nav Goal" button and draw a goal pose on the map. The robot will start to move towards that goal.
3. **View the navigation process:**
   See how the robot avoid obstacles during navigation.

Tip: Use different colors to distinguish between the global and local plans.

Navigate with a Known Map in Simulation

Congratulations if your robot has finished a navigation task!
You can dynamically change the navigation params using the rqt reconfigure tool.

Terminal New

```
$ rosrun rqt_reconfigure rqt_reconfigure
```

In the rqt reconfigure window, you can see several nodes on the left side. Select the “move base” node to see the related parameters.

Thanks for listening.

Codes and other materials are available on the Blackboard System.

Remember to return the sheet to the TAs before you leave.