ROS Navigation Stack

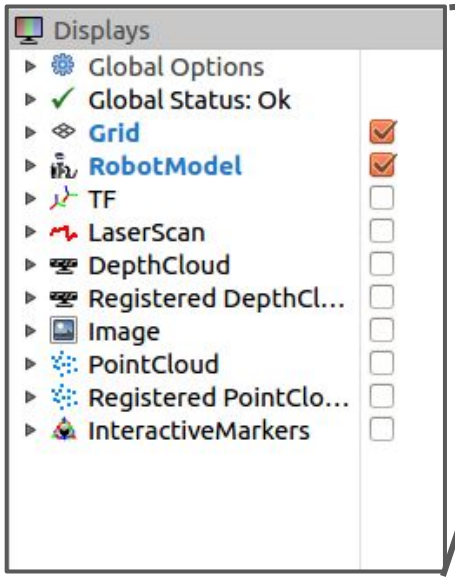
Movebase, gmapping, RViz

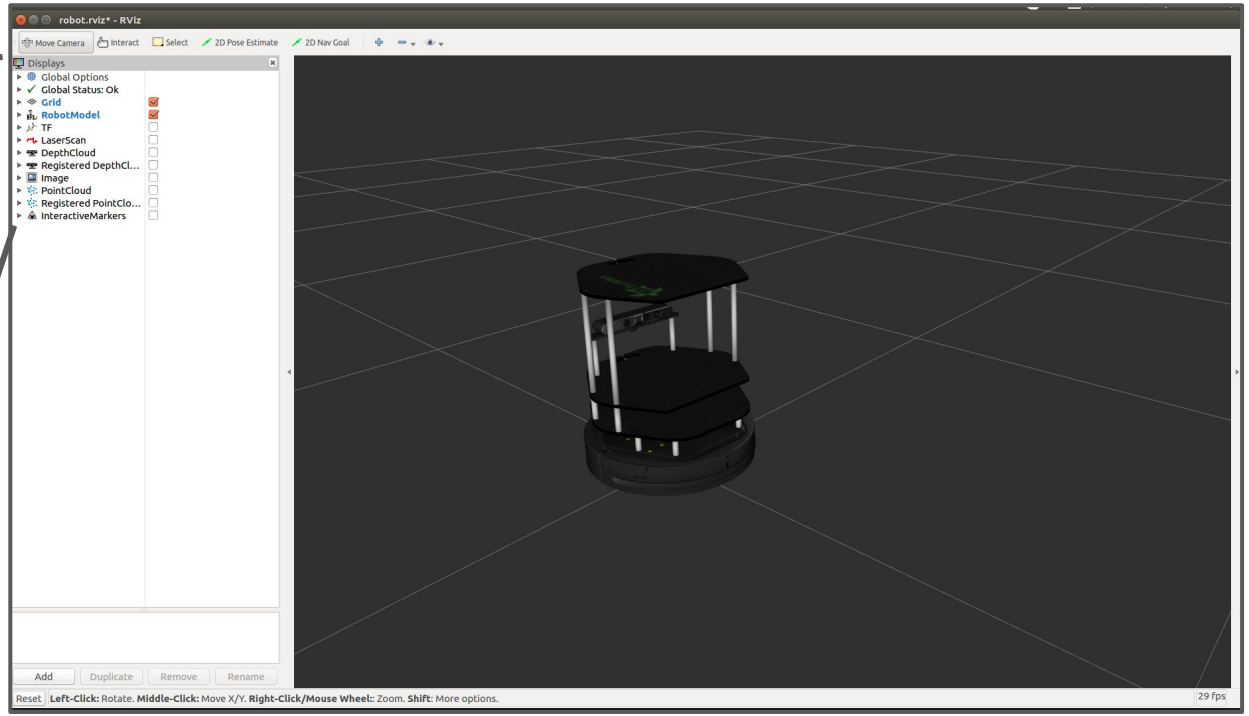
Robot Visualization - RViz

- 3D Visualization tool to view the world from the robot’s perspective
- Allows us to view other useful supplementary information about the robot
  Eg: Frames, Location, Path, etc.
- Rviz subscribes to ROS topics having a standard format
- Prebuilt visualization tools for several robot-related messages

Rviz with Turtlebot

```
roslaunch turtlebot_rviz_launchers view_robot.launch
```





Installing & Running

Installation:

```
sudo apt-get update
```

```
sudo apt-get install ros-kinetic-rviz
```

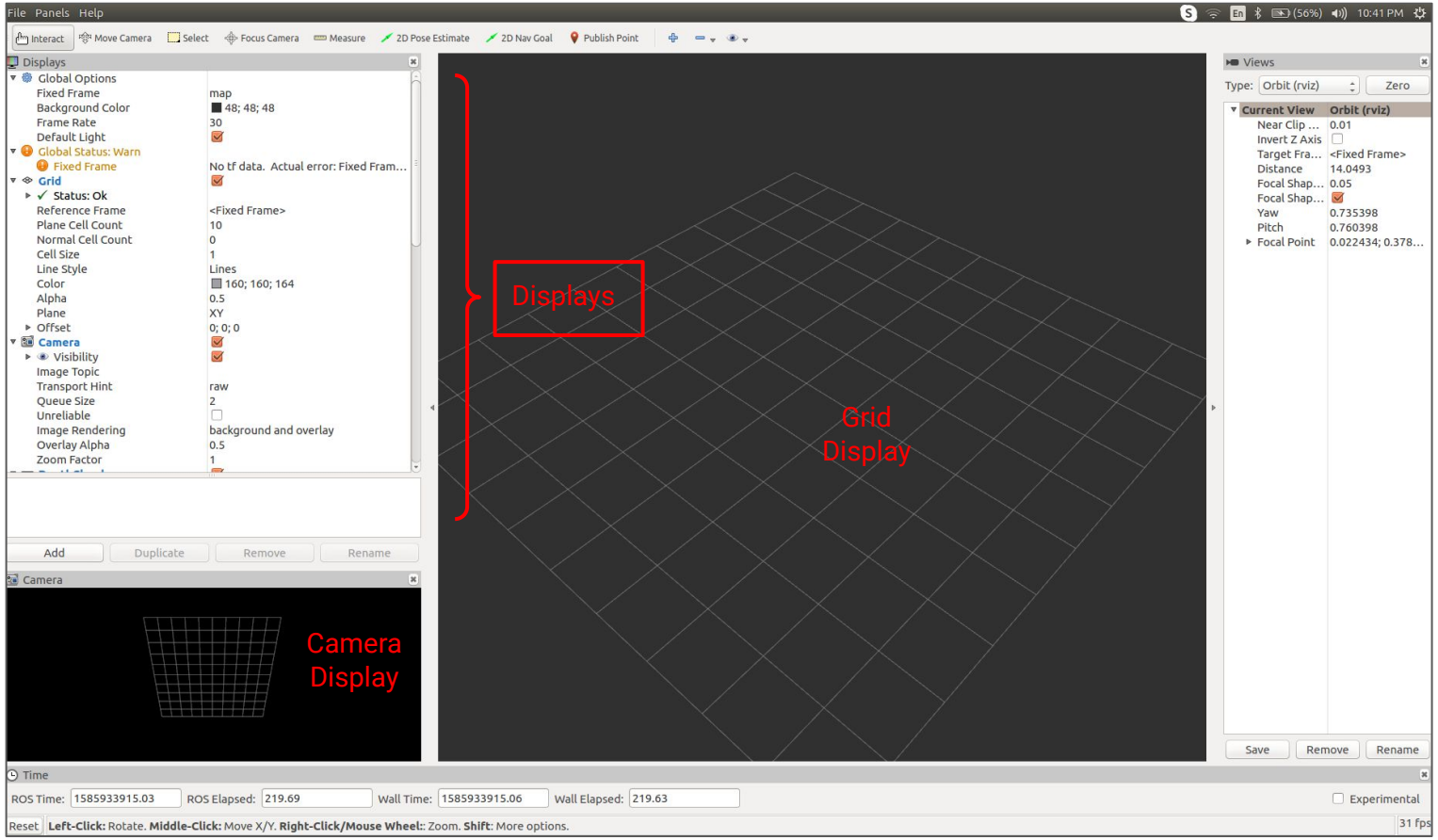
OR

```
sudo apt-get install ros-melodic-rviz
```

Running:

```
rosrun rviz rviz
```

RViz window


File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Default Light
Global Status: Warn
Fixed Frame No tf data. Actual error: Fixed Fram…
Grid
Status: Ok
Reference Frame <Fixed Frame>
Plane Cell Count 10
Normal Cell Count 0
Cell Size 1
Line Style Lines
Color 160; 160; 164
Alpha 0.5
Plane XY
Offset 0; 0; 0
Camera
Visibility
Image Topic
Transport Hint raw
Queue Size 2
Unreliable
Image Rendering background and overlay
Overlay Alpha 0.5
Zoom Factor 1
Add
Duplicate
Remove
Rename
Camera
Camera Display
Displays
Grid Display
Views
Type: Orbit (rviz)
Zero
Current View Orbit (rviz)
Near Clip … 0.01
Invert Z Axis
Target Fra… <Fixed Frame>
Distance 14.0493
Focal Shap… 0.05
Focal Shap…
Yaw 0.735398
Pitch 0.760398
Focal Point 0.022434; 0.378…
Save
Remove
Rename
Time
ROS Time: 1585933915.03
ROS Elapsed: 219.69
Wall Time: 1585933915.06
Wall Elapsed: 219.63
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps

Displays

Most common visualizable message types on ROS have a corresponding display option on RViz.

Display	Message type
Image	sensor_msgs/Image
LaserScan	sensor_msgs/LaserScan
Map	nav_msgs/OccupancyGrid
Path	nav_msgs/Path
Pose	geometry_msgs/PoseStamped
Odometry	nav_msgs/Odometry

Viewing Topics

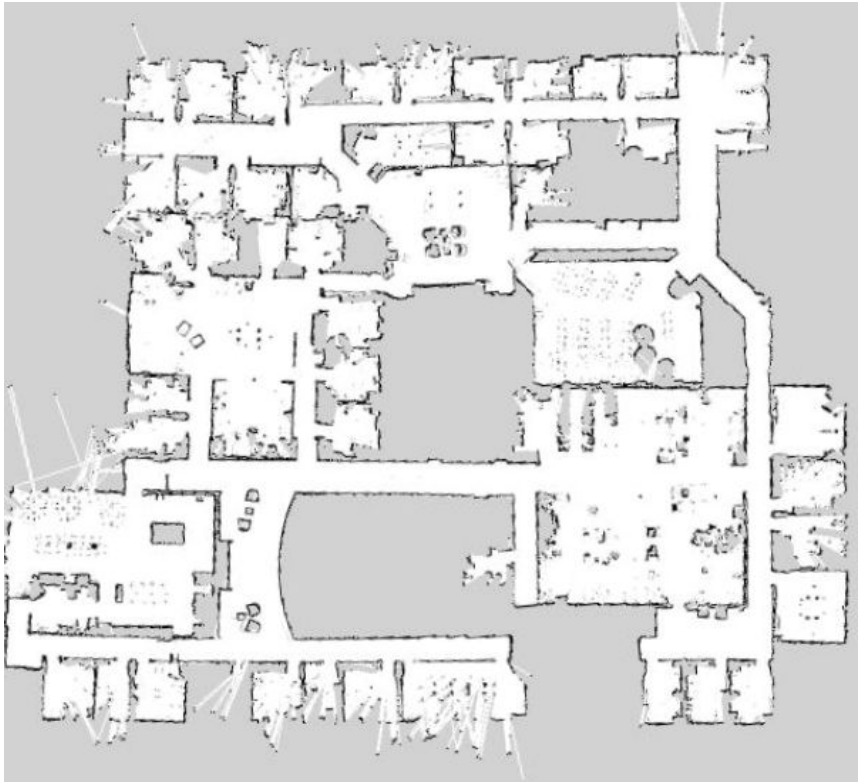
- In the ‘Displays’ menu, check the topics that you want to view
- To add a new topic, use the ‘Add’ option in the Displays menu
- Try to view:
  - LaserScan
  - Depth Map
  - Camera output
  - Transforms
  - Odometry

Building Maps in ROS

- Maps in ROS are stored as an **Occupancy Grid**

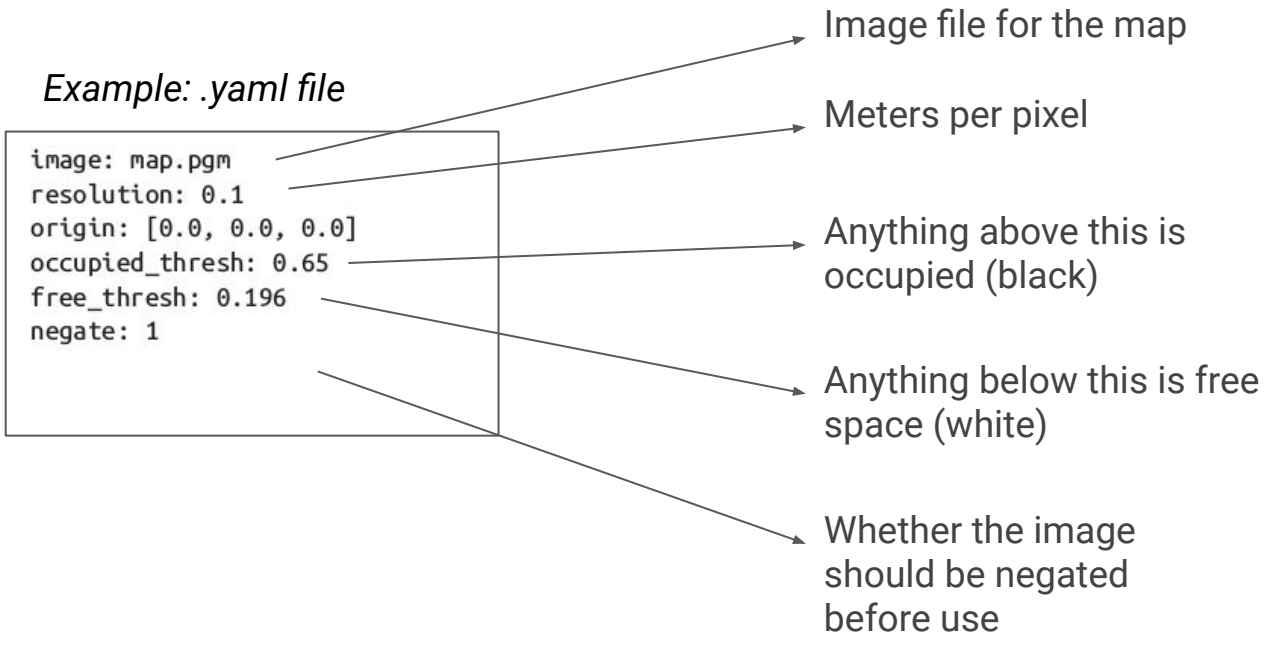
- White denotes free space
  Black denotes occupied space
  Grey denotes unmapped regions

- 2D maps of the world can be built in ROS using the package **gmapping.**



Map YAML Files

Maps are stored as a YAML* FIle that is linked to an image file (png, jpg, jpeg, etc.)



*Recursive Acronym - YAML stands for: YAML Ain't a Markup Language

Using gmapping

GMapping:

- SLAM Algorithm
- Uses a Particle Filter

Running:

1. Run turtlebot in Gazebo.
2. Run GMapping using: `rosrun gmapping slam_gmapping`
3. Launch Rviz and view the ‘Map’ display
4. Launch keyboard-teleop and move the robot around
5. Save the map using: `rosrun map_server map_saver`

AMCL (Adaptive Monte Carlo Localization)

- Probabilistic localization algorithms
- Uses a Map
- Compares laserscan with the map to correct pose estimate

AMCL Example:

1. Run turtlebot gazebo & view it in RViz
2. `roslaunch turtlebot_gazebo amcl_demo.launch`

ROS Navigation Stack

A fully developed navigation stack capable of giving odometry, planning paths and giving velocity commands to follow the paths

Turtlebot default Odometry: Sensor fusion (EKF) on IMU data and wheel encoders
ROS NavStack Odometry: Adaptive Monte Carlo using Laserscan.

Local Path Planner: Potential Field using *gradient method* on a ‘Cost Map’
Cost Map: A map denoting how ‘good’/’bad’ it is to be at a point
Eg: Being near obstacles is bad (high potential)
Gradient method: Field direction for planner is calculated by gradient of Cost Map

Move-base

Implementation of ROS actions to give velocity commands, given a destination Pose as an input.
Uses ROS navigation stack

Using move_base through rviz:

Use this to publish a goal-point through the GUI

View: Local path, cost map, odom, etc.

Sending Navigation Goals through Code

```python
import rospy
import actionlib

from move_base_msgs.msg import MoveBaseActoin, MoveBaseGoal

waypoint = {'position':[2.0. 2.0, 0.0], 'orientation':[0.0, 0.0, 0.0, 1.0]}

def goal_pose(pose):
    goal_pose = MoveBaseGoal()

    goal_pose.target_pose.header.frame_id = 'map'

    goal_pose.target_pose.pose.position.x = goal['position'][0]
    goal_pose.target_pose.pose.position.y = goal['position'][1]
    goal_pose.target_pose.pose.position.z = goal['position'][2]

    goal_pose.target_pose.pose.orientation.x = goal['orientation'][0]
    goal_pose.target_pose.pose.orientation.y = goal['orientation'][1]
    goal_pose.target_pose.pose.orientation.z = goal['orientation'][2]
    goal_pose.target_pose.pose.orientation.w = goal['orientation'][3]

    return goal_pose


if __name__ == '__main__':
    rospy.init_node('nav_goal')

    client = actionlib.SimpleActionClient('move_base', MoveBaseActoin)
    client.wait_for_server()

    while True:
        client.send_goal(goal_pose(waypoint))
        client.wait_for_result()
```

References for further learning

1. Morgan Quigley Chapter 9 (For mapping)
2. GMapping ROS Wiki: http://wiki.ros.org/gmapping
3. Morgan Quigley Chapter 10 (For Navigation Stack)
4. Move_base ROS Wiki: http://wiki.ros.org/move_base
5. NavStack: http://wiki.ros.org/navigation
6. Monte Carlo Localization: https://en.wikipedia.org/wiki/Monte_Carlo_localization