NAVIGATING WITH JACKAL

Below are the example launch files for three different configurations for navigating Jackal:

- Navigation in an odometric frame without a map, using only move_base.
- Generating a map using gmapping.
- Localization with a known map using amcl.

If you're using *simulation*, bring up Jackal with the front laser enabled for the following demos:

```
roslaunch jackal_gazebo jackal_world.launch config:=front_laser
```

If you're working with a real Jackal, it's suggested to connect via SSH and launch the jackal_navigation launchfiles from on board the robot. You'll need to have bidirectional communication with the robot's roscore in order to launch rviz on your workstation.

NAVIGATION WITHOUT A MAP

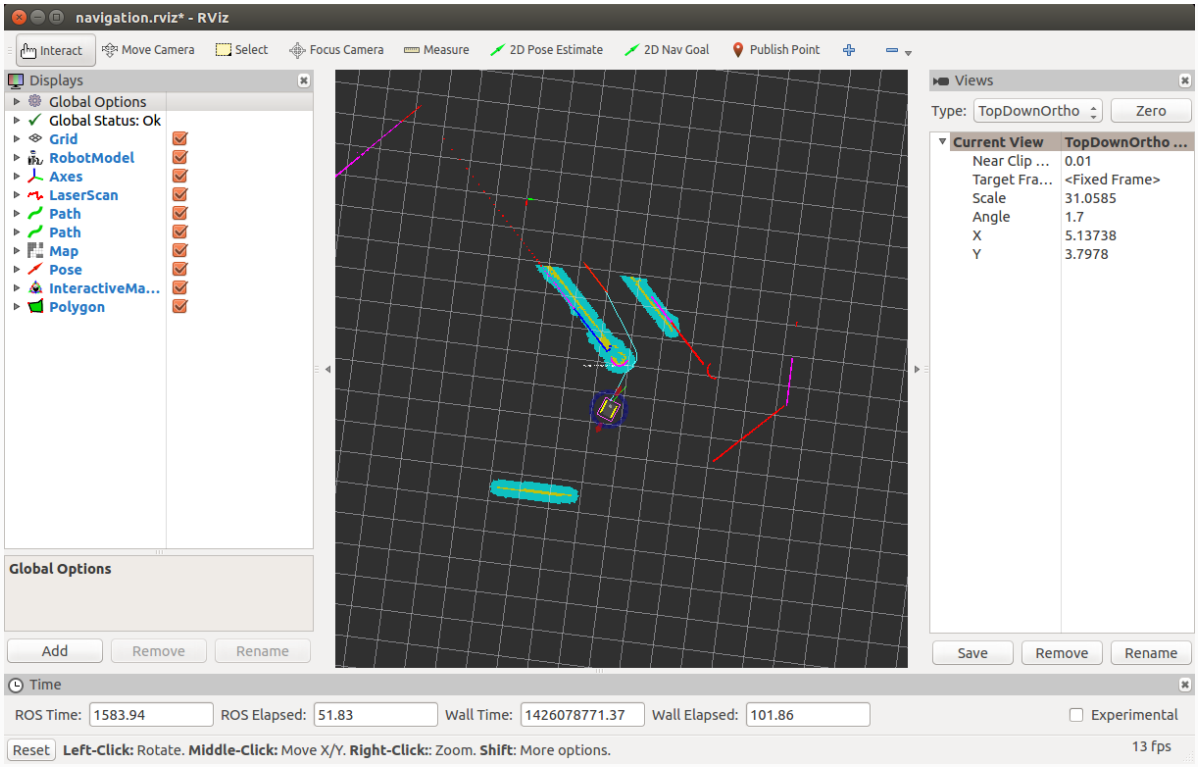
In the odometry navigation demo Jackal attempts to reach a given goal in the world within a user-specified tolerance. The 2D navigation, generated by move_base, takes in information from odometry, laser scanner, and a goal pose and outputs safe velocity commands. In this demo the configuration of move_base is set for navigation without a map in an odometric frame (that is, without reference to a map).

To launch the navigation demo, run:

```
roslaunch jackal_navigation odom_navigation_demo.launch
```

To visualize with the suggested rviz configuration launch:

```
roslaunch jackal_viz view_robot.launch config:=navigation
```



To send goals to the robot, select the *2D Nav Goal* tool from the top toolbar, and then click anywhere in the rviz view to set the position. Alternatively, click and drag slightly to set the goal position and orientation.

If you wish to customize the parameters of move_base, local costmap, global costmap and base_local_planner, clone jackal_navigation into your own workspace and modify the corresponding files in the *params* subfolder.

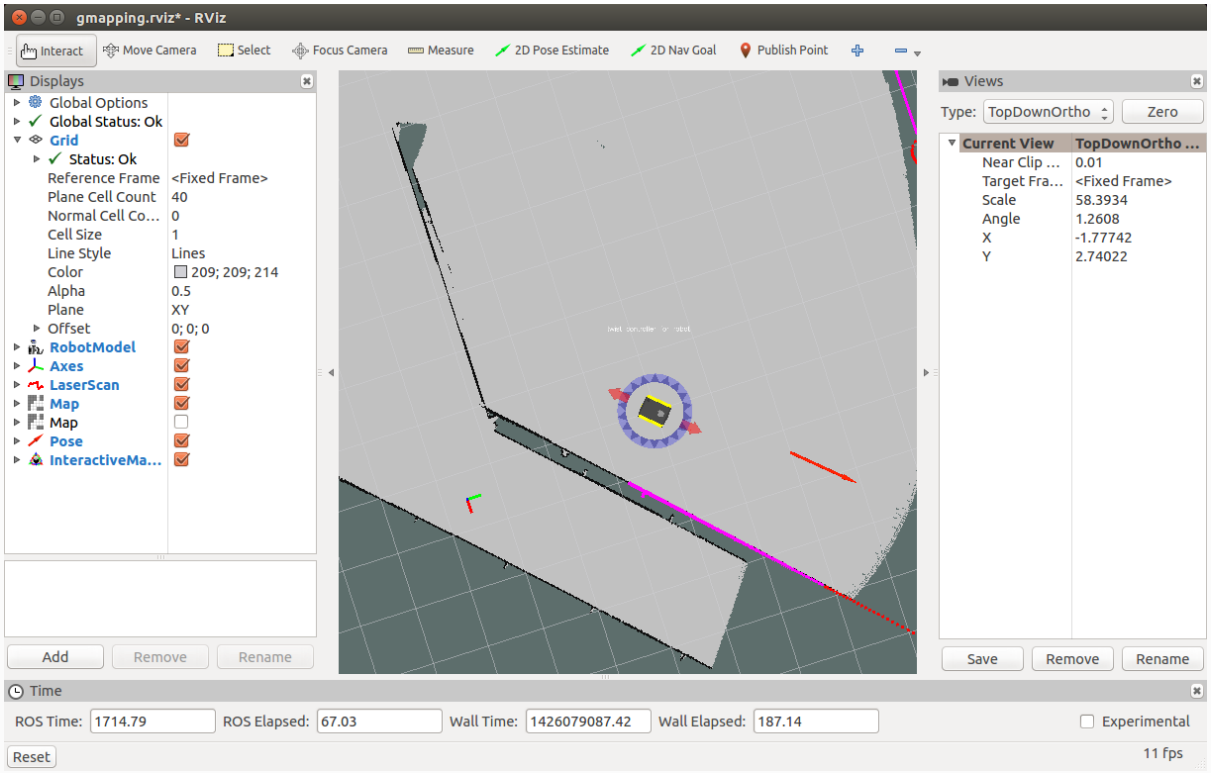
MAKING A MAP

In this demonstration, Jackal generates a map using gmapping. Begin by launch the gmapping launch file on the robot:

```
roslaunch jackal_navigation gmapping_demo.launch
```

And on your workstation, launch rviz with the suggested configuration:

```
roslaunch jackal_viz view_robot.launch config:=gmapping
```



You must slowly drive Jackal around to build the map. As obstacles come into view of the laser scanner, they will be added to the map, which is shown in rviz. You can either drive manually using the interactive markers, or semi-autonomously by sending navigation goals (as above).

When you're satisfied, you can save the produced map using map_saver:

```
rosrun map_server map_saver -f mymap
```

This will create a `mymap.yaml` and `mymap.pgm` file in your current directory.

NAVIGATION WITH A MAP

Using amcl, Jackal is able to globally localize itself in a known map. AMCL takes in information from odometry, laser scanner and an existing map and estimates the robot's pose.

To start the AMCL demo:

```
roslaunch jackal_navigation amcl_demo.launch map_file:=/path/to/my/map.yaml
```

If you don't specify `map_file`, it defaults to an included pre-made map of the default "Jackal Race" environment which Jackal's simulator spawns in. If you're using a real Jackal in your

own environment, you'll definitely want to override this with the map created using the gmapping demo.

Before navigating, you need to initialize the localization system by setting the pose of the robot in the map. This can be done using 2D Pose Estimate in rviz or by setting the amcl initial_pose parameters. To visualize with the suggested rviz configuration launch:

```
roslaunch jackal_viz view_robot.launch config:=localization
```

When rviz appears, select the *Set 2D Pose tool* from the toolbar, and click on the map to indicate to the robot approximately where it is.