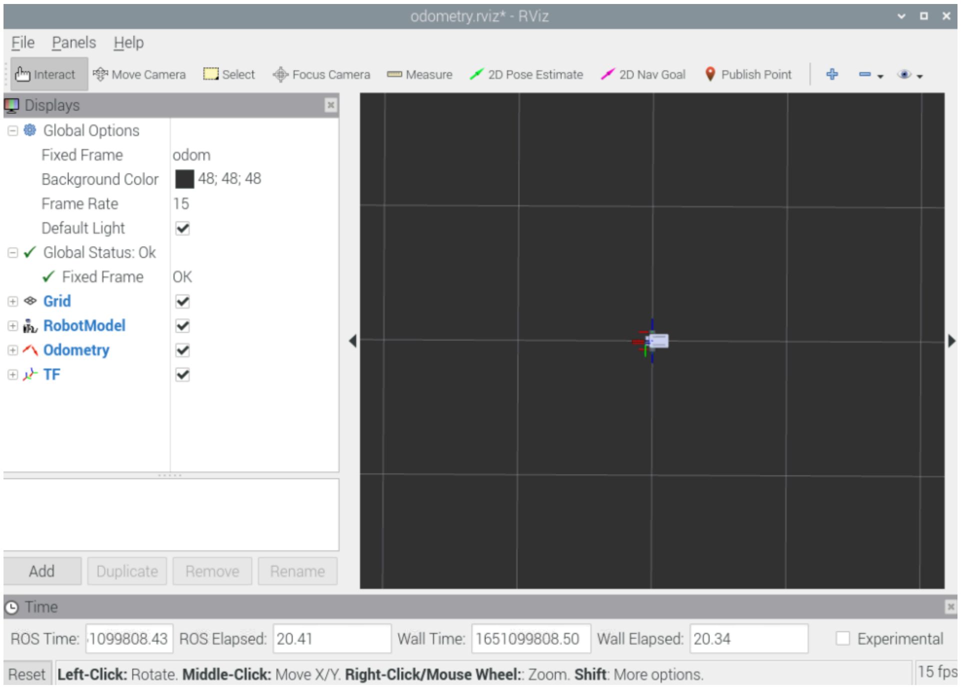
Week 5 Update: Rosify and Simple Autonomous Driving

4/29/22

ME/EE/CS 169

Tyler Nguyen and Lorenzo Shaikewitz

1 Update RVIZ

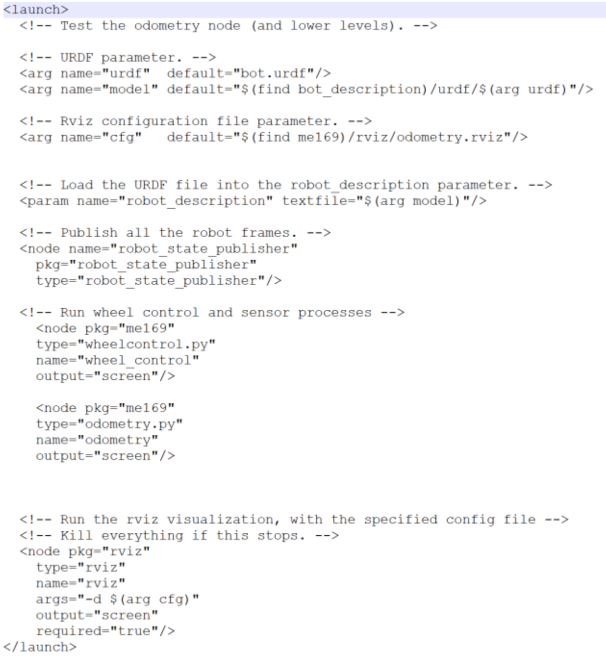

odometry.rviz* - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Add Duplicate Remove Rename
Time
Reset

Global Options
Fixed Frame: odom
Background Color: 48; 48; 48
Frame Rate: 15
Default Light: ☑
Global Status: Ok
✓ Fixed Frame: OK
Grid ☑
RobotModel ☑
Odometry ☑
TF ☑

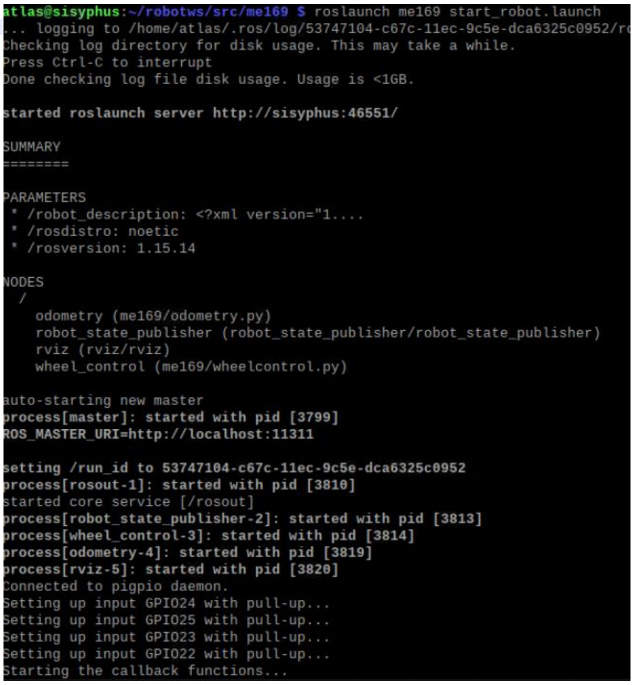
ROS Time: 1099808.43 ROS Elapsed: 20.41 Wall Time: 1651099808.50 Wall Elapsed: 20.34 □ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 15 fps

2 Roslaunch



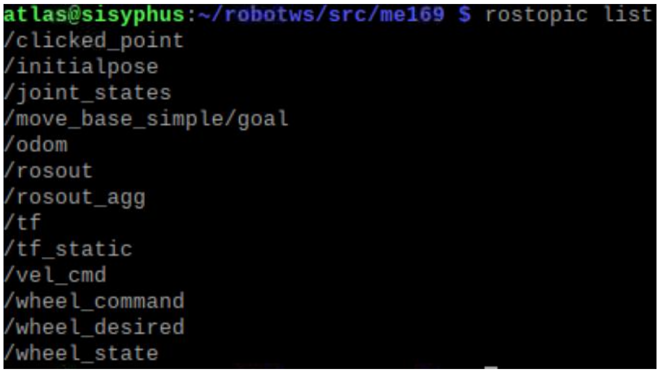
```
<launch>
  <!-- Test the odometry node (and lower levels). -->

  <!-- URDF parameter. -->
  <arg name="urdf"  default="bot.urdf"/>
  <arg name="model" default="$(find bot_description)/urdf/$(arg urdf)"/>

  <!-- Rviz configuration file parameter. -->
  <arg name="cfg"   default="$(find me169)/rviz/odometry.rviz"/>


  <!-- Load the URDF file into the robot_description parameter. -->
  <param name="robot_description" textfile="$(arg model)"/>

  <!-- Publish all the robot frames. -->
  <node name="robot_state_publisher"
    pkg="robot_state_publisher"
    type="robot_state_publisher"/>

  <!-- Run wheel control and sensor processes -->
    <node pkg="me169"
    type="wheelcontrol.py"
    name="wheel_control"
    output="screen"/>

    <node pkg="me169"
    type="odometry.py"
    name="odometry"
    output="screen"/>



  <!-- Run the rviz visualization, with the specified config file -->
  <!-- Kill everything if this stops. -->
  <node pkg="rviz"
    type="rviz"
    name="rviz"
    args="-d $(arg cfg)"
    output="screen"
    required="true"/>
</launch>
```



```
atlas@sisyphus:~/robotws/src/me169 $ roslaunch me169 start_robot.launch
... logging to /home/atlas/.ros/log/53747104-c67c-11ec-9c5e-dca6325c0952/ro
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://sisyphus:46551/

SUMMARY
========

PARAMETERS
 * /robot_description: <?xml version="1....
 * /rosdistro: noetic
 * /rosversion: 1.15.14

NODES
  /
    odometry (me169/odometry.py)
    robot_state_publisher (robot_state_publisher/robot_state_publisher)
    rviz (rviz/rviz)
    wheel_control (me169/wheelcontrol.py)

auto-starting new master
process[master]: started with pid [3799]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 53747104-c67c-11ec-9c5e-dca6325c0952
process[rosout-1]: started with pid [3810]
started core service [/rosout]
process[robot_state_publisher-2]: started with pid [3813]
process[wheel_control-3]: started with pid [3814]
process[odometry-4]: started with pid [3819]
process[rviz-5]: started with pid [3820]
Connected to pigpio daemon.
Setting up input GPIO24 with pull-up...
Setting up input GPIO25 with pull-up...
Setting up input GPIO23 with pull-up...
Setting up input GPIO22 with pull-up...
Starting the callback functions...
```

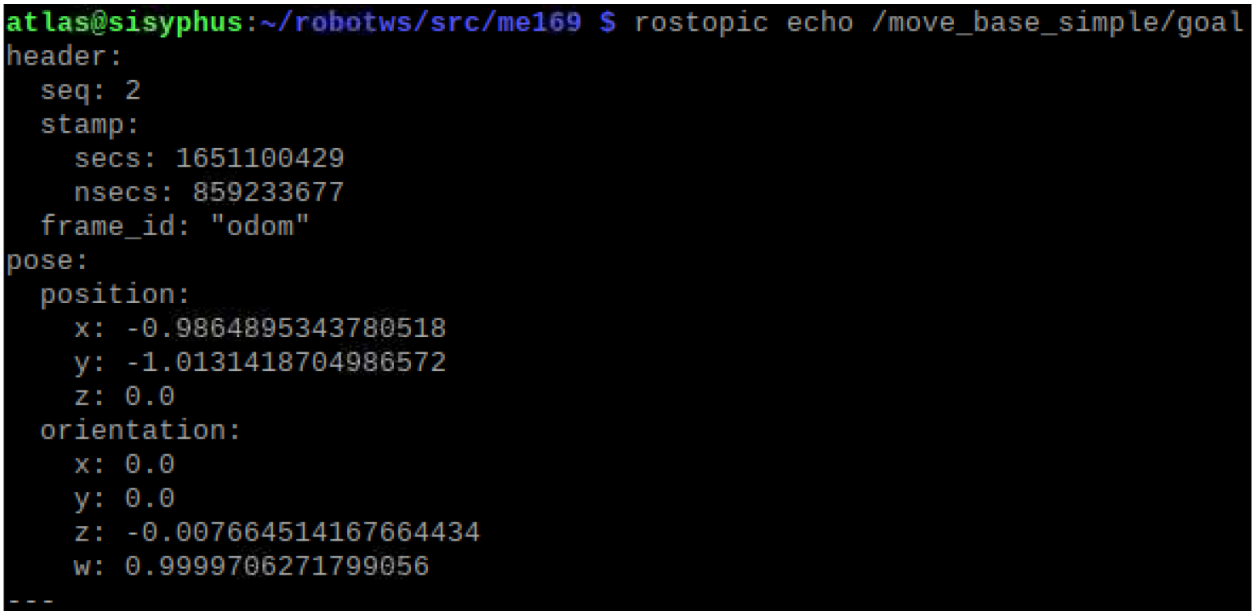
3 2D Navigation Goal

When running RVIZ, the list of ROS topics contains the topic /move_base_simple/goal, which provides information on the 2D Navigation Goal set through the RVIZ GUI.



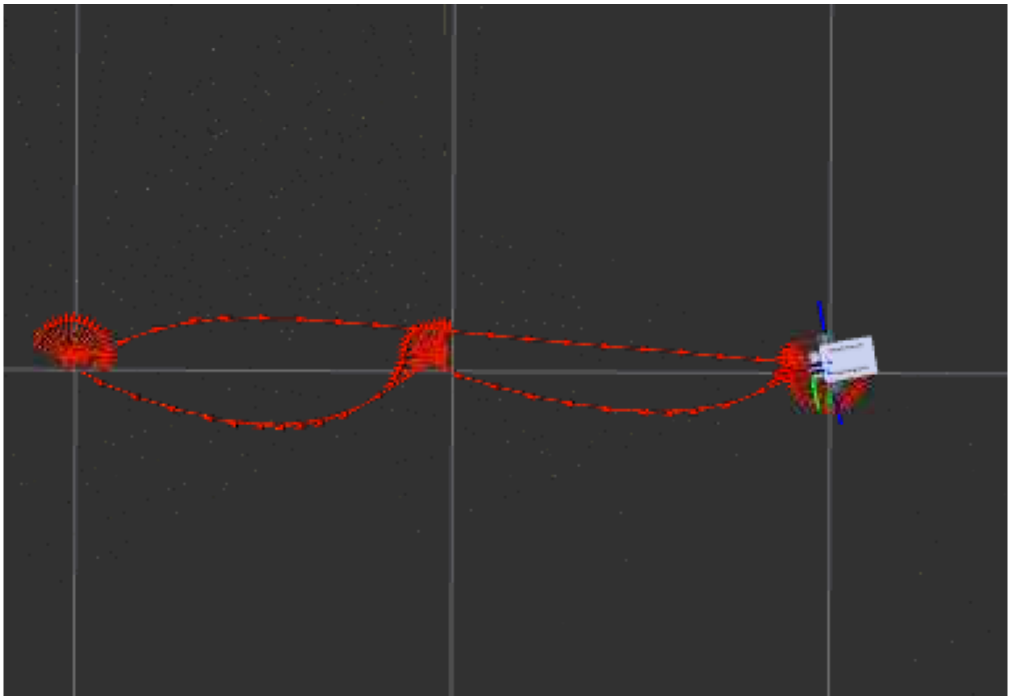
```
atlas@sisyphus:~/robotws/src/me169 $ rostopic list
/clicked_point
/initialpose
/joint_states
/move_base_simple/goal
/odom
/rosout
/rosout_agg
/tf
/tf_static
/vel_cmd
/wheel_command
/wheel_desired
/wheel_state
```

Echoing the topic allows us to view the topic message when we set a 2D Navigation Goal in RVIZ. We note that the pose is given in the odom frame, as desired. This frame changes dependent on RVIZ's reference frame.

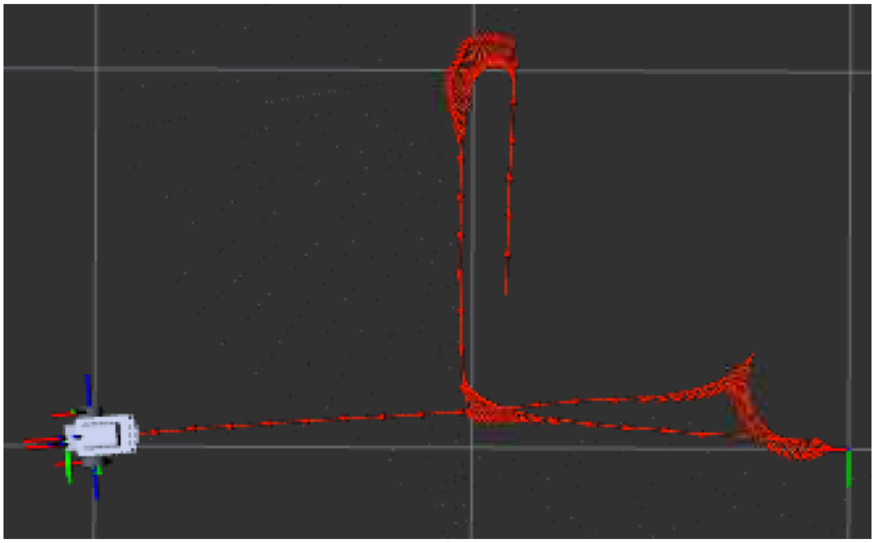


```
atlas@sisyphus:~/robotws/src/me169 $ rostopic echo /move_base_simple/goal
header:
  seq: 2
  stamp:
    secs: 1651100429
    nsecs: 859233677
  frame_id: "odom"
pose:
  position:
    x: -0.9864895343780518
    y: -1.0131418704986572
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: -0.007664514167664434
    w: 0.9999706271799056
---
```

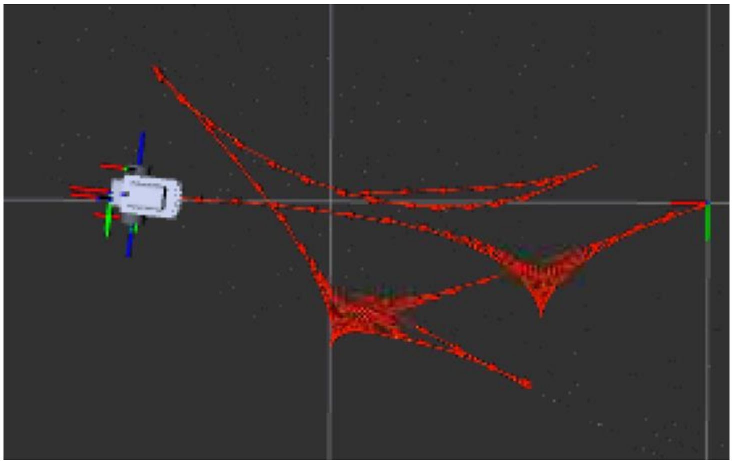
4 Create a Simple Autonomous Driving Node



For a simple autonomous driving node, we first utilized a turn-drive-turn method. This method included a proportional speed based upon the distance to the goal, thus slowing down as it reached the provided navigational goal. It also included a deadzone such that when close enough to the goal it would switch to the turning phase.

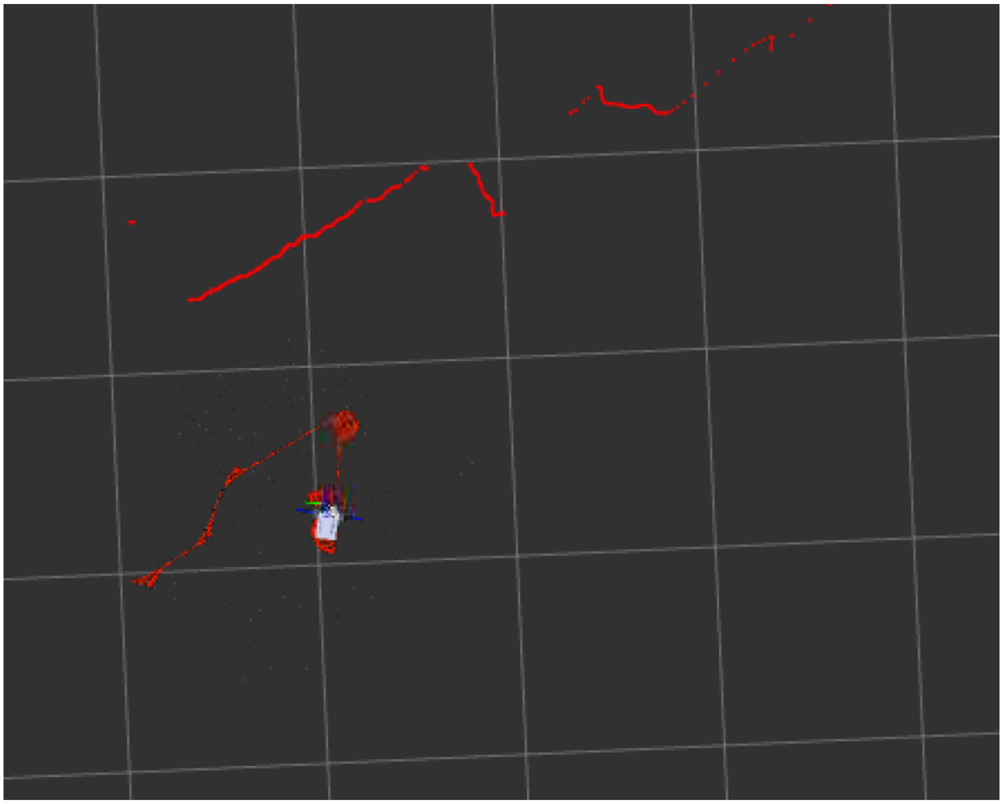


A second method we utilized consisted of a turn and drive, then orient to the desired orientation. It also slows down as it arrives to the goal.



Another method utilizes the same turn and drive, then orientation method as before, but now utilizes a trigonometric feedback rather than a linear feedback for the angular velocity while driving to the goal. This appeared much smoother than the previous method.

5 Add the Laser Scanner



The laser scan works well! We can even detect Lorenzo's feet when he jumps in front of the robot to scare Tyler. We added the laser scanner to the launch file and added the LaserScan display to the rviz configuration.