Project: Were Am I

Abstract:

The project is to create ROS package, develop a mobile robot model for Gazebo, and integrate Adaptive Monte Carlo Localization (AMCL) and Navigation package for localizing the robot in the provided map and move to target.

Introduction:

There are couple of steps in the project.

1. Create ROS package

First, create a directory catkin_me for the project.

$mkdir -p ~/catkin_me/src

$cd ~/catkin_me/src

Second, create package udacity_bot under it.

$catkin_create_pkg udacity_bot

$ cd udacity_bot

Then create directory for worlds and launch.

$ mkdir launch

$ mkdir worlds

Copy world description from project instruction to udacity.world under the worlds directory. Udacity.world is an XML file contains ground plane, light source, and world camera for the Gazebo environment.

The launch directory contains launch files for the project. As you can see under ~/catkin_me/src/udacity_bot/launch directory, there are three launch files, amcl.launch, robot_description.launch, and udacity_world.launch. Udacity_world.launch will launch a robot defined by robot_description.launch and an empty world with jackal_race.world.

2. Develop a mobile robot model for Gazebo

We can use Unified Robot Description Format (URDF) to develop a robot in the simulation environment. For this project udacity_bot.xacro defines the robot in ~/catkin_me/src/udacity_bot/urdf. There are links, footprint, chassis, left_wheel, right_wheel, camera, and hokuyo (laser). Robot_footprint_joint connects chassis to robot_footprint as fixed. Joint left_wheel_hinge connects link left_wheel to chassis as continuous mode as wheel can rotate. Similarly, right_wheel_hinge connects to chassis as continuous mode. Camera_joint connects camera to chassis as fixed. hokuyo_joint connects laser to chassis as fixed.

Inside definition of a link, it might contain inertial, collision, and visual definitions. For example, left_wheel has mass five units, and 3x3 rotational matrix only diagonal elements with one. In collision properties, the geometry is cylinder with radius 0.1 meter and length 0.05 meter. Material visual property specifies it with green color.

```
  <link name='left_wheel'>
```

```
    <inertial>
      <mass value="5.0"/>
      <origin xyz="0.0 0 0" rpy=" 0 1.5707 1.5707"/>
      <inertia
          ixx="0.1" ixy="0" ixz="0"
          iyy="0.1" iyz="0"
          izz="0.1" />
    </inertial>
    <collision name='collision'>
      <origin xyz="0 0 0" rpy=" 0 1.5707 1.5707"/>
      <geometry>
        <cylinder radius=".1" length="0.05"/>
      </geometry>
    </collision>
    <visual name='left_wheel_visual'>
      <origin xyz="0 0 0" rpy=" 0 1.5707 1.5707"/>
      <geometry>
        <cylinder radius=".1" length="0.05"/>
      </geometry>
      <material name="green">
            <color rgba="0 1.0 0 1.0" />
      </material>
    </visual>
  </link>
```

For Hokuyo laser scanner, it has an image mesh from ”package://udacity_bot/meshes/hokuyo.dae”. Fig. 1 shows a screenshot of robot in RViz.

Under ~/catkin_me/src/udacity_bot/urdf/udacity_bot.gazebo, it defines Gazebo plugin. Libgazebo_ros_diff_drive.so is a C++ shared library.

```
<gazebo>
```

```
    <plugin name="differential_drive_controller"
filename="libgazebo_ros_diff_drive.so">
      <legacyMode>false</legacyMode>
      <alwaysOn>true</alwaysOn>
      <updateRate>10</updateRate>
      <leftJoint>left_wheel_hinge</leftJoint>
      <rightJoint>right_wheel_hinge</rightJoint>
      <wheelSeparation>0.4</wheelSeparation>
      <wheelDiameter>0.2</wheelDiameter>
      <torque>10</torque>
      <commandTopic>cmd_vel</commandTopic>
      <odometryTopic>odom</odometryTopic>
      <odometryFrame>odom</odometryFrame>
      <robotBaseFrame>robot_footprint</robotBaseFrame>
    </plugin>
  </gazebo>
```

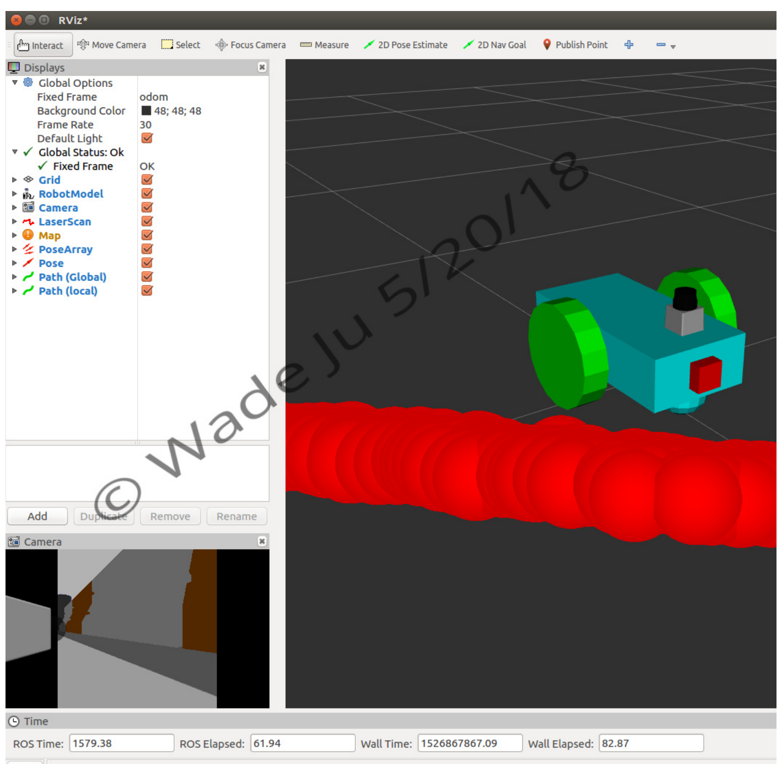

Fig. 1 Robot on RViz

3. AMCL and Navigation Packages

Then we need to setup map with AMCL for localization and Navigation package to move robot around obstacles to the target.

By coping jackal_race.pgm and jackal_race.yaml from https://github.com/udacity/RoboND-Localization-Project/tree/master/maps to ~/catkin_me/src/udacity_bot/urdf/maps. And set argument world_name in ~/catkin_me/src/udacity_bot/launch/udacity_world.launch to jackal_race.world.

Adaptive Monte Carlo Localization (AMCL) dynamically adjusts the number of particles over a period of time, as the robot navigates around in a map. This adaptive process offers a significant computational advantage over MCL.

We create amcl.launch under ~/catkin_me/src/udacity_bot/launch. It contains map server, localization, laser, odom, and move base configurations. For move base configuration, there are four configuration files under directory ~/catkin_me/src/udacity_bot/config.

```
Let’s launch it with the following command.

$ roslaunch udacity_bot udacity_world.launch

In a new terminal, launch AMCL model as well.
$ roslaunch udacity_bot amcl.launch

The you can see Gazebo as Fig.2 and RViz as Fig. 3
```

<!-- Map server —>

```
  <arg name="map_file" default="$(find udacity_bot)/maps/jackal_race.yaml"/>
  <node name="map_server" pkg="map_server" type="map_server" args="$(arg
map_file)" />
 <!-- Localization-->
 <node pkg="amcl" type="amcl" name="amcl" output="screen">
   <remap from="scan" to="udacity_bot/laser/scan"/>
   <param name="odom_frame_id" value="odom"/>
   <param name="odom_model_type" value="diff-corrected"/>
   <param name="base_frame_id" value="robot_footprint"/>
   <param name="global_frame_id" value="map"/>
   <param name="min_particles" value="10"/>
   <param name="max_particles" value="80"/>
   <param name="kid_err" value="0.001"/>
   <param name="update_min_d" value="0.25"/>
   <param name="update_min_a" value="0.25"/> <!-- pi/12 -->
   <param name="resample_interval" value="1"/>

   <param name="transform_tolerance" value="0.01"/>
   <!-- laser -->
   <param name="laser_model_type" value="likelihood_field" />
   <param name="laser_z_hit" value="0.8"/>
   <param name="laser_z_rand" value="0.2"/>
   <param name="laser_max_beams" value="50"/>
   <param name="laser_max_range" value="12" />
   <param name="laser_likelihood_max_dist" value="1.5" />

   <!-- odometry -->
   <param name="odom_alpha1" value="0.02"/> <!-- 0.05 -->
   <param name="odom_alpha2" value="0.02"/>
   <param name="odom_alpha3" value="0.02"/>
   <param name="odom_alpha4" value="0.02"/>
   <!-- <param name="odom_alpha5" value="0.2"/> -->
 </node>
<!-- Move base -->
 <node pkg="move_base" type="move_base" respawn="false"
name="move_base" output="screen">
   <rosparam file="$(find udacity_bot)/config/costmap_common_params.yaml"
command="load" ns="global_costmap" />
   <rosparam file="$(find udacity_bot)/config/costmap_common_params.yaml"
command="load" ns="local_costmap" />
```

```
    <rosparam file="$(find udacity_bot)/config/local_costmap_params.yaml"
command="load" />
    <rosparam file="$(find udacity_bot)/config/global_costmap_params.yaml"
command="load" />
    <rosparam file="$(find udacity_bot)/config/base_local_planner_params.yaml"
command="load" />

    <remap from="cmd_vel" to="cmd_vel"/>
    <remap from="odom" to="odom"/>
    <remap from="scan" to="udacity_bot/laser/scan"/>

    <param name="base_global_planner" type="string" value="navfn/NavfnROS" /
>
    <param name="base_local_planner" value="base_local_planner/
TrajectoryPlannerROS"/>

  </node>
```

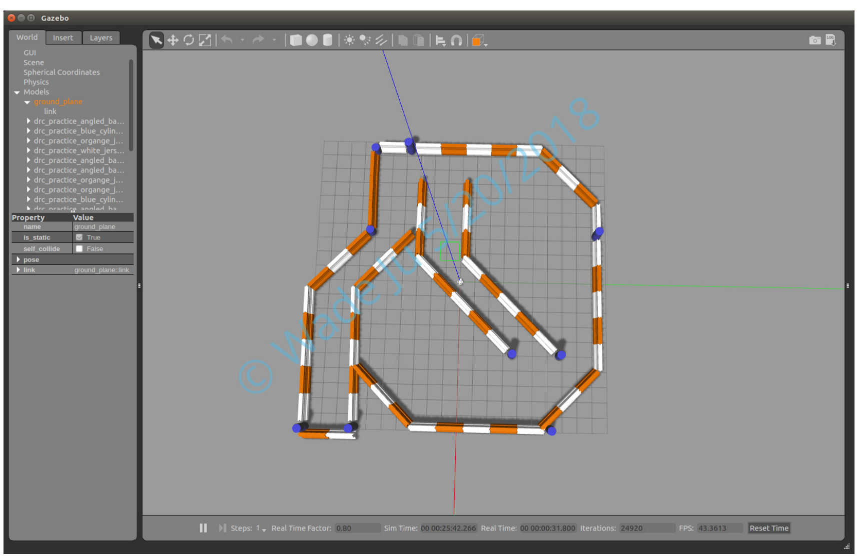

Fig. 2 Gazebo shows race track

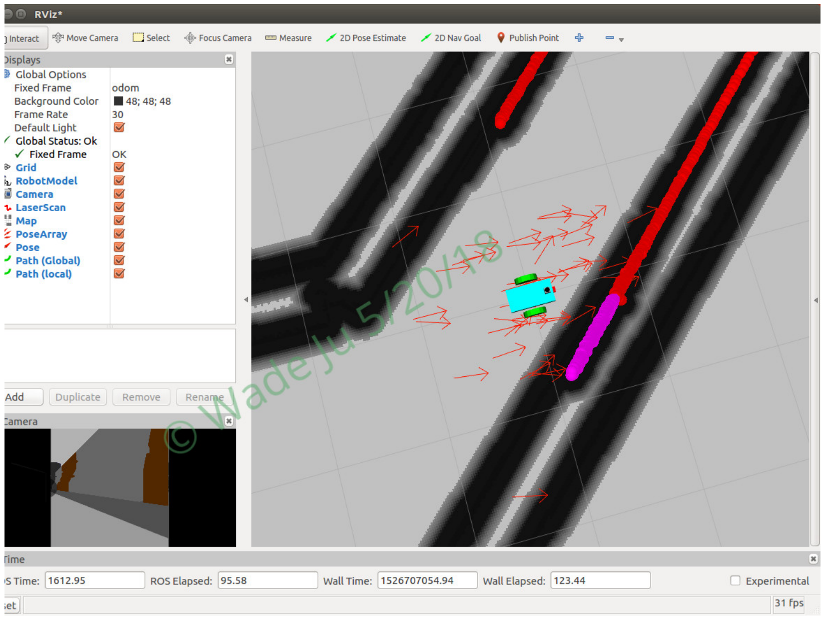

Fig. 3 RViz shows Robot in Race Track

Base_local_planner_params.yaml defines parameters for TrajectoryPlannerROS including (x,y, rotational) acceleration limitation, maximum and minimum (x,y, rotational) velocity, holonomic or non-holonomic robot, goal tolerance parameters, trajectory scoring parameters, forward simulation parameters, etc.

costmap_common_params.yaml defines common parameters for both global and local cos-maps for example obstacle_range for obstacle thresholds in the cos-map . Here are properties that set in costmap_common_params.yaml.

obstacle_range: 2.5
raytrace_range: 3.0
transform_tolerance: 0.3
robot_radius: 0.1
inflation_radius: 0.2
observation_sources: laser_scan_sensor
laser_scan_sensor: {sensor_frame: hokuyo, data_type: LaserScan, topic: /udacity_bot/laser/scan, marking: true, clearing: true}

Global_costmap_params and local_costmap_params define properties for global and local respectively. global_costmap_params sets global frame to map. local_costmap_params sets global_frame to odom. Both set update_frequency and publish_frequency to 10. It would be to set them to 20Hz as default control frequency. However, my Ubuntu machine can’t keep up computation

Results: Navigation command sends commend to Robot to reach goal. By issuing the command on a new terminal,

$rosrun undacity_bot navigation

Robot has reached goal as shown in Fig. 4 and Fig. 5.

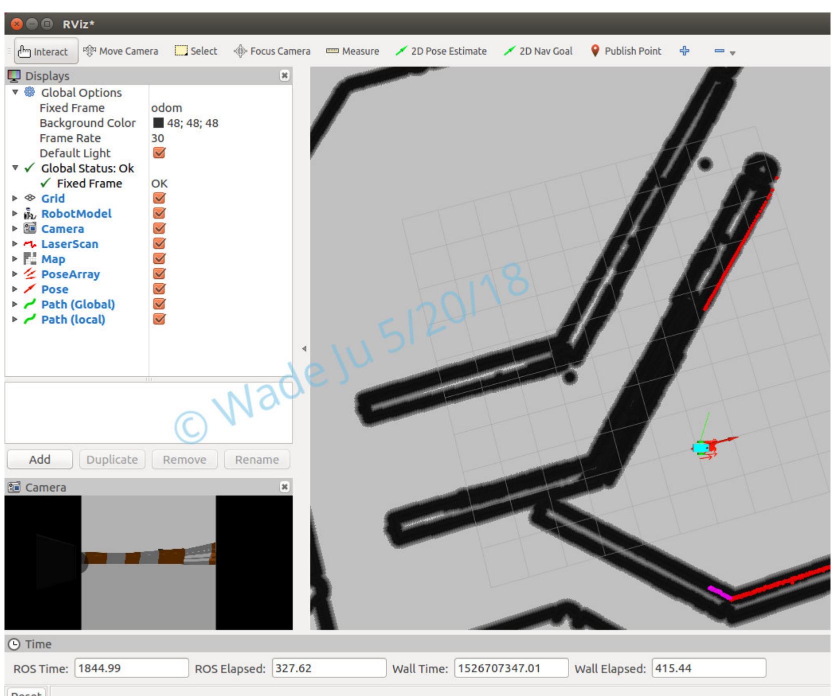

Fig. 4 Robot Reaches Goal (RViz)

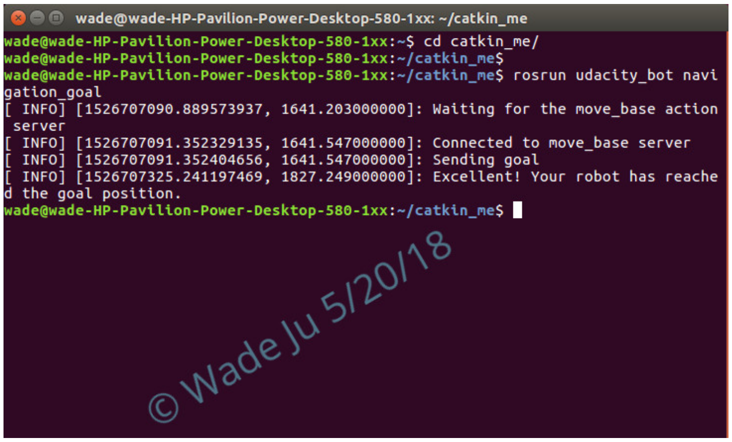

Fig. 5 Robot Reaches Goal (Terminal)

Discussion: To run AMCL and navigation stack take quite a lot computation power as well as rendering on Gazebo and RViz. When I was working on Ubuntu VM (VMWare) on Mac, I got bogus screen like Fig. 6. Later on, I bought a PC and installed Ubuntu 16 and installed ROS packages. Then things went back to normal.

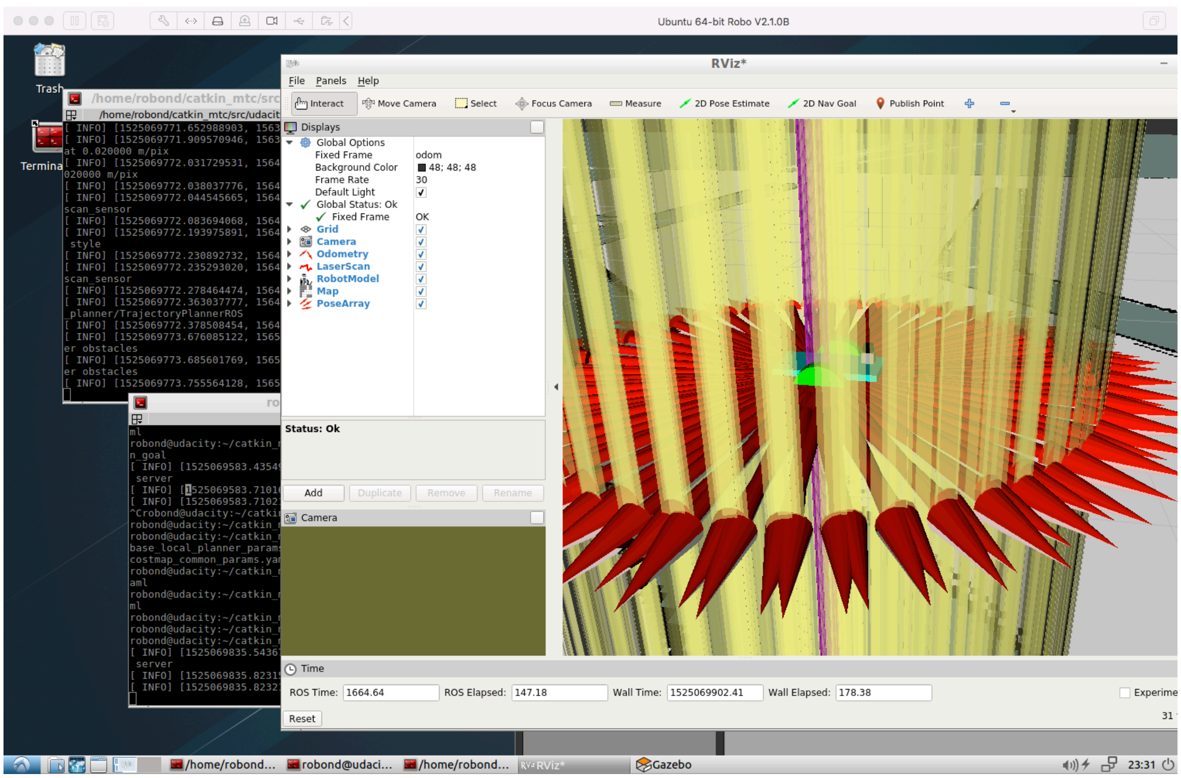

Fig. 6 RViz Rendering problem on VM

Conclusion / Future Work: Only camera and laser are used in the project for localization and navigation. Future work can add more sensors like IMU, RADAR, infra-red, etc. Computation capability is also an important factor. An AMD Ryzen 1700 CPU with 16 GB memory native Ubuntu 16.04 can't fully support the system with 20Hz in Gazebo and rViz.

References

1. http://osrf-distributions.s3.amazonaws.com/sdformat/api/dev.html
2. http://wiki.ros.org/navigation/Tutorials
3. http://wiki.ros.org/amcl

2. http://wiki.ros.org/navigation