19AIE213 ROBOTIC OPERATING SYSTEMS AND ROBOT SIMULATIONS

Project Report

Submission Date: **08-06-2021**

ROBOTIC OPERATING SYSTEMS FOR MAZE NAVIGATIONS

Name	Roll Number
Adarsh Shankar R S	AM.EN.U4AIE19003
Adithya D	AM.EN.U4AIE19005
Akash M	AM.EN.U4AIE19007
Hrishikesh P	AM.EN.U4AIE19033
Madhav M Menath	AM.EN.U4AIE19039
Nithin Sylesh	AM.EN.U4AIE19044
Samuel Joshua V	AM.EN.U4AIE19056
V.Nikhil	AM.EN.U4AIE19067
Vysakh S Nair	AM.EN.U4AIE19072
Yadukrishnan J	AM.EN.U4AIE19073

1. Project Title:

ROBOTIC OPERATING SYSTEMS FOR MAZE NAVIGATIONS

Video simulation in the below link:
https://drive.google.com/drive/folders/1QLFdHlNNkuivcZMKRGrNld2tNxXStXRJ?usp=sharing

2. Abstract:

Preliminary project on creating a map of a virtual world and then use it to navigate around. Basically we navigate a map in a simulated world.

A video showing how to develop the simulation is shared. We need a world to create a simulation in Gazebo

The *Robot Operating System* (*ROS*) sets up an ideal platform of software libraries and tools in building up the robot we desire.As a part of this project, we look to add necessary sensors corresponding to Camera and LADAR. In terms of simulation, we look to actually move the built robot using Teleop key and we ensure it is set for further visualisation. This way we look to progressively solve the provided task efficiently.

3. Libraries and Packages Used :

The desired package contains a C++ parser for the Unified Robot Description Format (URDF), which is an XML format for representing a robot model. The code API of the parser has been through our review process and will remain backwards compatible in future releases. Specific package tends to contain ROS nodes and a ROS-independent library. Implemented code provides a pretty evident overview on the packages and libraries used. The package contains a number of XML specifications for robot models, sensors, scenes.

Plugin
Gazebo plugins give your URDF models greater functionality and can tie in ROS messages and service calls for sensor output and motor input.

4. Detailed Project Description：

The way of approach in regard to solving the task provided is as follows:

1. Editing URDF file to add 3 sensors - LADAR, Camera
2. Move the built robot with teleop key
3. Viewing output in rviz & gazebo

Before actually quickly surging into step(1), it is vita that we have a proper understanding of URDF. The Unified Robotic Description Format (URDF) is an XML file format used in ROS to describe all elements of a robot. It could also be claimed as a collection of files that describe a robot's physical description to ROS. To use a URDF file in Gazebo, some additional simulation-specific tags must be added to work properly with Gazebo.

Gazebo is basically a 3D simulator and with Gazebo we are able to create a 3D scenario on our computer with robots, obstacles and many other objects. RVIZ is another 3D visualization tool useful for ROS applications and it is capable of displaying data from cameras, lasers and corresponding 3D devices.
In our case we view the obtained output using both of them and observe it.It is also important to understand that Gazebo supports several plugin types, and all of them can be connected to ROS, although only a few types can be referenced through a URDF file. And we can add plugins to Gazebo if we desire.

Notably, we also look to interface the Hokuyo laser as a part of this model. This particular laser stops publishing scans during the test, which takes about a minute. The result of the test is in the response returned by this service.

The Teleop key acts pretty much like a controller. We actually use the arrow keys to give linear x and angular z commands on the output topic.

We desire to visualize the model built using RVIZ. At the bottom of the 'Displays' group, you'll find buttons 'Add', 'Remove', 'Rename' for adding items to visualize, removing items and renaming them. Let's first add the robot model, so click 'Add' and scroll to 'rviz > RobotModel' and click 'OK'. It should now look like the display below. Also, it is further possible to view sensor information using RVIZ. This visualization is also possible in Gazebo.

5. Simulation Results:

Camera plugin

```
    <sensor type="camera" name="camera1">
        <update_rate>30.0</update_rate>
        <camera name="head">
            <horizontal_fov>1.3962634</horizontal_fov>
            <image>
                <width>800</width>
                <height>800</height>
                <format>R8G8B8</format>
            </image>
            <clip>
                <near>0.02</near>
                <far>300</far>
            </clip>
            <noise>
                <type>gaussian</type>
                <mean>0.0</mean>
                <stddev>0.007</stddev>
            </noise>
        </camera>
    <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
       <alwaysOn>true</alwaysOn>
       <updateRate>0.0</updateRate>
       <cameraName>rrbot/camera1</cameraName>
       <imageTopicName>image_raw</imageTopicName>
       <cameraInfoTopicName>camera_info</cameraInfoTopicName>
       <frameName>laser_scanner</frameName>
       <hackBaseline>0.07</hackBaseline>
       <distortionK1>0.0</distortionK1>
       <distortionK2>0.0</distortionK2>
       <distortionK3>0.0</distortionK3>
       <distortionT1>0.0</distortionT1>
       <distortionT2>0.0</distortionT2>
   </plugin>
</sensor>
```

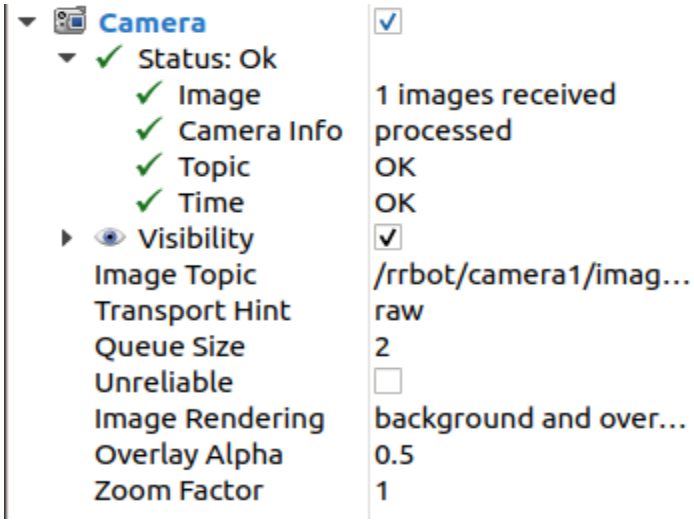
Differential Drive Plugin

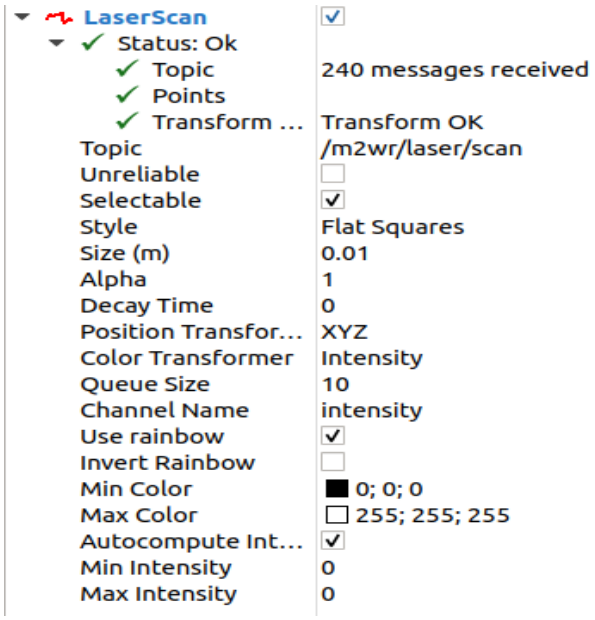
```
<plugin name="differential_drive_controller" filename="libgazebo_ros_diff_drive.so">
 <alwaysOn>true</alwaysOn>
 <updateRate>20</updateRate>
 <leftJoint>joint_left_wheel</leftJoint>
 <rightJoint>joint_right_wheel</rightJoint>
 <wheelSeparation>0.25</wheelSeparation>
 <wheelDiameter>0.07</wheelDiameter>
 <torque>20</torque>
 <commandTopic>cmd_vel</commandTopic>
 <odometryTopic>odom</odometryTopic>
 <odometryFrame>odom</odometryFrame>
 <robotBaseFrame>chassis</robotBaseFrame>
 </plugin>
```

Laser Plugin

```
    <sensor type="ray" name="head_hokuyo_sensor">
        <pose>0 0 0 0 0 0</pose>
        <visualize>true</visualize>
        <update_rate>20</update_rate>
        <ray>
            <scan>
                <horizontal>
                    <samples>720</samples>
                    <resolution>1</resolution>
                    <min_angle>-1.570796</min_angle>
                    <max_angle>1.570796</max_angle>
                </horizontal>
            </scan>
            <range>
                <min>0.10</min>
                <max>10.0</max>
                <resolution>0.01</resolution>
            </range>
            <noise>
                <type>gaussian</type>
                <mean>0.0</mean>
                <stddev>0.01</stddev>
            </noise>
        </ray>
    <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_laser.so">
        <topicName>/m2wr/laser/scan</topicName>
        <frameName>laser_scanner</frameName>
    </plugin>
</sensor>
```

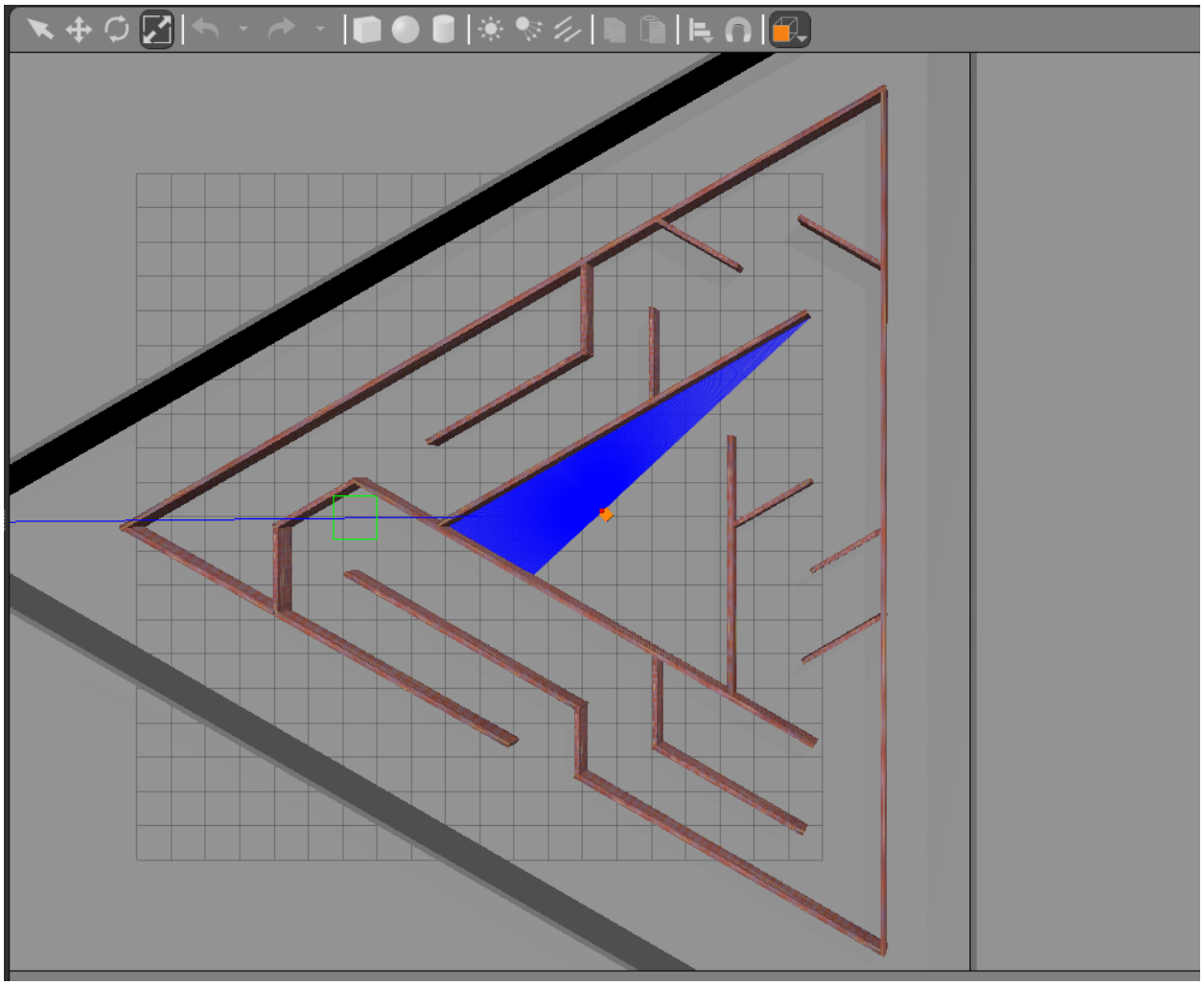
Plugin Stats in Rviz:



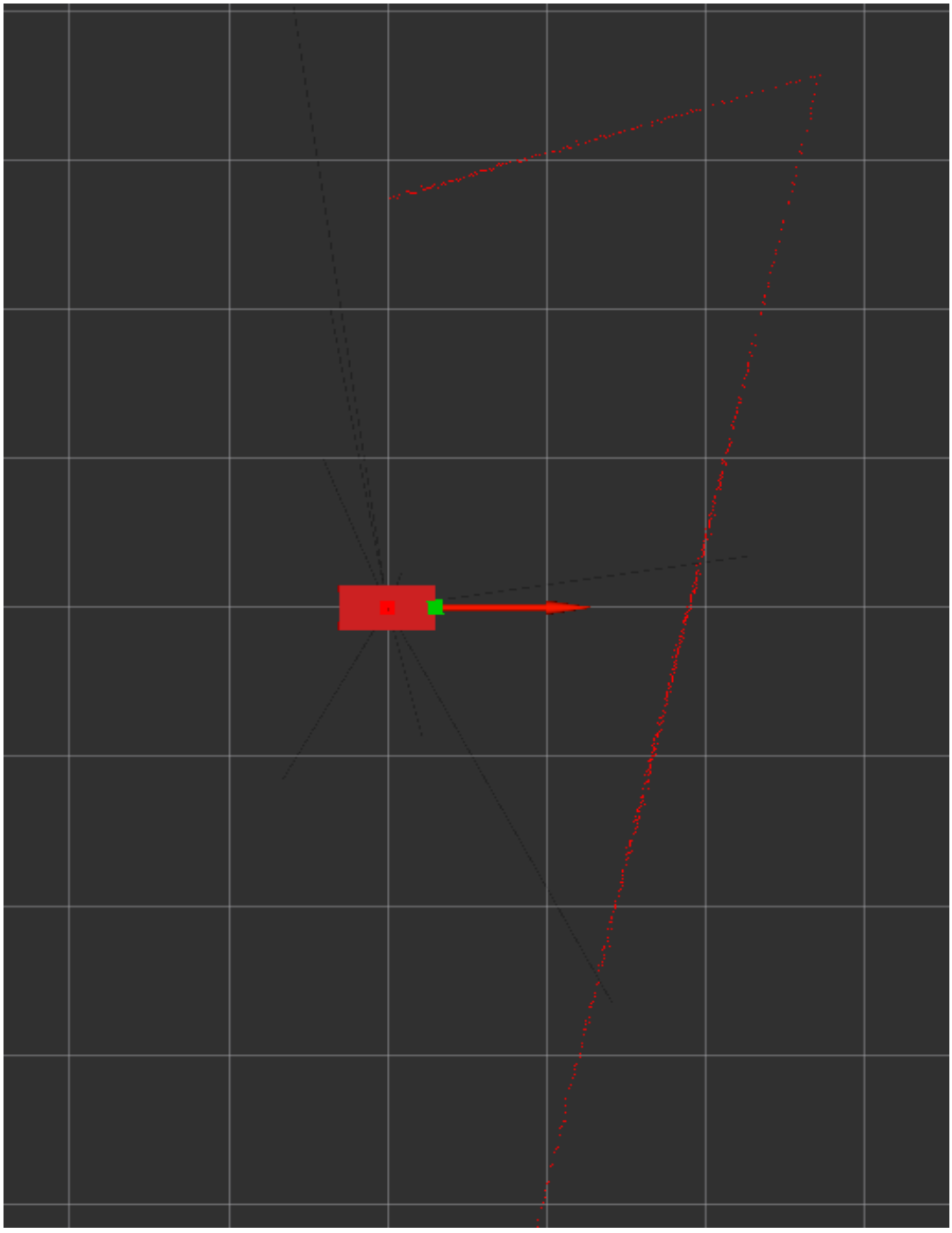


Odometry	☑
Status: Ok	
Topic	96 messages received
Transform ...	Transform OK
Topic	/odom
Unreliable	☐
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	100
Shape	Arrow
Covariance	☑
Position	☑
Orientation	☐

Simulation in Gazebo



Simulation in Rviz



NOTE: All the code and video of the simulation can be found here: