Week 5 Lab – Simulating Robot Localization

In the previous lab and challenge we focused on sensor fusion based on recordings of real data. In this lab we will move into simulation and explore the localisation problem.

This lab and the next week's lab will provide you with your groups' submission to the second challenge. We strongly advise that you complete this lab's material before next week to avoid falling behind.

NOTE: Remember to follow the instructions on SurreyLearn to source the packages not installed on your computer!

NOTE: It is highly recommended you do this lab on otter machines. **Gazebo is not stable on WSL**.

Exercise 1: Running Gazebo

This exercise will guide you through getting started with Gazebo, creating worlds and spawning robots.

1. **Getting Started:** Create a new package in your workspace called lab5. Next, start a roscore and open gazebo using the launch file option:

```
$ roslaunch gazebo_ros empty_world.launch
```

2. **Get Familiar with Gazebo:** Click the insert tab in the top left and drag out some models to create an interesting scene. Remember that these models may take a few seconds to download and will take up some room in your home space. Play around for and make sure you are happy with the interface (remember left click moves, middle click rotates and right click zooms).
3. **Build an Environment:** Next we will create an environment for a Turtlebot. When we spawn the Turtlebot, it will appear at the origin so make sure it stays clear. Add a few models to the environment, try to create an interesting scene to explore. Click "File > Save World As" and save the world as "lab5_world" in the "worlds" directory of your package (create it if it doesn't exist). Close Gazebo.
4. **Start Gazebo with your World:** The "empty_world.launch" file has an argument called "world_name", use it to launch gazebo with the custom world you created, once you are happy your world has been opened, close Gazebo. **Note: You will have to use the FULL path to your world file!**

```
$ roslaunch gazebo_ros empty_world.launch world_name:=<path_to_your_world>
```

5. **Turtlebot Setup:** Add the line

```
export TURTLEBOT3_MODEL=burger
```

to your "~/.bashrc" file, you can use the command

```
$ echo "export TURTLEBOT3_MODEL=burger" >> ~/.bashrc
```

to do so.

6. **Spawn a Turtlebot:** The Turtlebot is well integrated into ROS and Gazebo and therefore has a custom launch file. It exists in the “turtlebot3_gazebo” package and is called “turtlebot3_empty_world.launch”. Launch this file (note it will not yet load your world).
7. **Move the Turtlebot around:** Launch the “turtlebot3_teleop_key.launch” file from the “turtlebot3_teleop” package. Make sure you can move the Turtlebot around in the simulator.
8. **Set up RVIZ:** Start RVIZ using the command:

```
$ roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch
```

Since we haven’t localised the Turtlebot yet, you will have to set the frame_id of RVIZ to “odom”.

Why is that the case?
Who provides the TF map<-Odom?

Your visualisation should include: TF, Odometry and LaserScan. Make sure you can visualise all topics.

9. **Customise the Launch File**: Copy the “tutlebot3_world.launch” file to your package, in the “launch” folder and call it “ex1.launch”. Inspect the launch file and find out how it is launching Gazebo (*Hint: it uses the same launch file that we did*). Customize it to use your world by default, launch rviz and use a custom rviz config file. *Hint: add “export EDITOR=gedit” to your .bashrc and use rosed and roscd*.
10. **Understand the Launch File**: Spend some time understanding the launch file you have just copied.

How does the URDF file get set?
What is missing?
Why do we need “turtlebot3_gazebo_rviz.launch”?

Hint: Where is the “robot_state_publisher”?

11. **Unified Launch file:** Add a “robot_state_publisher” and an rviz to your launch file.

Exercise 2: Localization

In this exercise we will use a “map” of the world to localise.

1. **Get Data:** From SurreyLearn, download the ex2.world gazebo file along with the ex2_map.yaml and ex2_map.pgm files. Store the world file in the “worlds” directory, and the maps in a new “maps” directory.
2. **Make a new Launch File:** Create a copy of your ex1.launch file called ex2.launch. Change the default world to point ex2.world. Add two additional nodes: an “amcl” node and a “map_server” node. Make sure the map_server uses your new ex2_map.yaml file, and that no extra topics need to be remapped.
3. **Run your launch file**: In RVIZ display the particle cloud, the estimated pose and the map (as in the lecture demonstration).
4. **Send Pose Estimate**: Use the “2D Pose Estimate” button in RVIZ to send an initial pose to AMCL. Make sure you click then drag to select the correct position and orientation.
5. **Teleoperate the robot via the keyboard:** Observe how the particle filter converges over time and how this depends on the environment and the robot’s movement. Note that the quality of the convergence may depends on the scene you have set up in Gazebo.
6. **Explore Parameters:** Explore how the parameters of the amcl node effect performance. In particular, the population size and initial distribution described during the lecture. Also explore how the LiDAR plugin parameters affect performance, particularly the noise and maximum distance parameters.
7. **Kidnapped Robot:** Assume your robot is now lost. Call the “/global_localization” service. What happens? Can you think of ways to improve performance?

Exercise 3: Navigation

In this exercise we will use everything we have done to start the ROS navigation stack. We will then use the navigation stack to move the Turtlebot around, perform pathplanning and avoid obstacles.

1. **Add Navigation Stack:** Create a copy of your ex2.launch file called ex3.launch. Add a new line that includes the "move_base" launch file from the "turtlebot3_navigation" package:

```
<include file="$(find turtlebot3_navigation)/launch/move_base.launch">
  <arg name="model" value="$(arg model)" />
  <arg name="move_forward_only" value="false"/>
</include>
```

   Launch the file.
2. **Visualise in RVIZ:** If your launch file ran successfully, you should be able to visualise everything in RVIZ. First, make two RVIZ groups called "Global" and "Local" by using the "Add" button. Click add again and start adding the topics under "/move_base" in the "By Topic" tab. When you add a topic, drag it into the corresponding Group. In the Global group, add the Global Costmap, Global Path and NavFn Path. In the Local Group, add the Local Path, the Local Costmap (make sure you colour this one using the "costmap" palette) and the local Footprint. The topic names should be enough indication of where each of these elements is. If in doubt, use the launch file as a reference (and look at the parameter YAML files).
3. **Send Waypoints from RVIZ:** Localise the Turtlebot like in the previous section (with a 2D pose estimate and a bit of teleoperating). Use the "2D Nav Goal" button in RVIZ to send a goal to the Turtlebot. Make sure you kill the Teleop launch file, as otherwise the Turtlebot won't move. Can you explain why that is? Look at the first two lines of the "move_base" launch file. Keep sending waypoints.
4. **Send Waypoint from Code:** The 2D Nav Goal button sends a message out on the topic "/move_base_simple/pose". Use the terminal commands to figure out what message type this is. Once you have done this, write a Python or C++ script that uses this message type to send a single goal to the Turtlebot.
5. **Do something Cool:** Try sending more complex paths for the Turtlebot to follow, can you time things so that the Turtlebot makes an 8 shape? What about collision avoidance? Adapt your code to subscribe to the current pose of the Turtlebot. Once you have this you can start to send goals after the Turtlebot has reached its destination!
6. **Explore Move Base:** If you haven't already done so, make a local copy of move_base launch file. Explore the parameters of each planner, and try out other global and local planners from here: http://wiki.ros.org/navigation

Exercise 4: Modifying robot definition

In this exercise, we will use URDF, XACRO, Gazebo and Launch files to modify the model that gets launched in Gazebo by adding a camera sensor.

1. **Make a new Launch File:** Make a new copy of ex3.launch and call it ex4.launch.
2. **Copy the URDF file:** Instead of simply including the Turtlebot URDF as we did last excercise, we are going to make our own copy. In your lab5 package, make a folder called "urdf". Copy the files "turtlebot3_burger.urdf.xacro" and "turtlebot3_burger.gazebo.xacro" from turtlebot3_description/urdf into the "urdf" folder in your lab5 package. *Hint: Use roscd to find where turtlebot3_description is.*
3. **Edit the Turtlebot Launch file:** Make a copy of "ex3.launch" and call it "ex4.launch" edit the file to use the local version of your urdf. **Hint:** Look at "robot_description" and how the xacro file is compiled! You may also look at last week's lecture. Launch the file to make sure it still works (and is using the version you think it is). You will also have to edit the urdf to use the local version of "turtlebot3_burger.gazebo.xacro".
4. **Create a Gazebo Plugin file:** In your lab5 package, create a file "turtlebot3_burger_camera.urdf.xacro" located at lab5/urdf/turtlebot3_burger_camera.urdf.xacro. Use provided file to create a new "ros_camera" gazebo plugin. **Remember**: you have to set the Links, Joints and Visual elements for this plugin (you can find a 3D mesh for the sensor on SurreyLearn) as well as the actual gazebo plugin. Use the parameters shown in the lecture. Launch the simulator.
5. **Edit the Xacro file:** Open the turtlebot3_burger.urdf.xacro and include your new Gazebo plugin. Use the lecture slides as reference if you need to. ***Hint:*** *Look at how the other plugins are being included!*
6. **Run the new Model:** Launch using your custom ex2.launch file. You should now see a red box on the turtlebot. Run RVIZ and visualise the output from the camera.
7. **Teleoperate the Robot:** Move the robot around using the teleop launch file. See what the sensors output.

Make sure you have finished this lab before next week, where we will be using everything we did here to explore and localise within environments we don't have a map for!