Robot Practical Course Bachelor

Assignment #3

This assignment introduces ROS' navigation software stack. You will create a map of the TAMS floor using Simultaneous Localization and Mapping (SLAM) techniques, and use this map for autonomous navigation. You should use this map throughout the rest of this course.

Task 3.1 Build a map: Often autonomous navigation in an environment requires an explicit map. This is used to estimate the current pose of the robot and to plan paths to desired goals.

3.1.1: Create your own map of the TAMS floor using a Simultaneous Localization and Mapping (SLAM) algorithm available in ROS. Include the lab F-326, and the hallway up to the fire door in front of the kitchen.

Hint: Open the doors in the environment completely to facilitate localization.

Start the turtlebot, remove it from the dock, connect your local ros network to the robot and run the teleoperation node as you did in the previous assignment.

Now you can run the following command on your development computer to start the SLAM node. The SLAM will be running on the development as has more computational power than the turtlebots raspberry pi. You can also run a strip down version of the SLAM node on the turtlebot, but it is not necessary for this assignment.

```
ros2 launch turtlebot4_navigation slam.launch.py
```

To visualize the progress of the mapping, you can use the RViz tool. Open a new terminal on your local computer and run the following command:

```
ros2 launch turtlebot4_viz view_robot.launch.py
```

Now drive the turtlebot around the environment using the teleoperation node. You can add an additional window to the RViz display to visualize the camera image for improved teleoperation.

Drive around the environment and make sure to cover all areas. Keep an eye on the RViz display to see how the map is built, all walls and static obstacles should be defined cleanly in the map. Dynamic obstacles such as humans or other turtlebots should not be included in the map.

Do not pick up the turtlebot while it is mapping, as this will lead to a wrong map.

3.1.2: Once you created a map of the environment, make sure to save it before killing the SLAM node! The saved map consists of an image and a yaml file. The map should represent the usual state of the environment and should not include dynamic obstacles such as humans or other turtlebots. Use an image editor (e.g. gimp) to clear such obstacles from your map.

You can save the map by running the following command in your development environment:

```
ros2 service call /slam_toolbox/save_map slam_toolbox/srv/SaveMap \
 "{name: {data: tams_floor}}"
```

Task 3.2 Autonomous navigation: Now that you have a map of the environment, you can use it to localize the robot in the environment. Additionally, you can use the map to plan paths to desired goals.

3.2.1: Start the localization stack on the turtlebot by running the following command on the turtlebot:

```
ros2 launch turtlebot4_navigation localization.launch.py \
  map:=<PATH_TO_YAML_FILE_OF_YOUR_MAP>
```

Replace `<PATH_TO_YAML_FILE_OF_YOUR_MAP>` with the path to the yaml file of your map. The yaml file should be in the same folder as the image file of your map. You need to deploy the map to the turtlebot first using the command you learned earlier otherwise the file will not be found.

3.2.2: Start the visualization tool RViz with some basic configuration on your local computer. Run the following command in a terminal to start RViz with the turtlebot navigation configuration:

```
ros2 launch turtlebot4_viz view_robot.launch.py
```

Click on the 2D pose estimate button in the RViz toolbar and click on a point in the map where the turtlebot is located. While clicking drag the mouse in the direction the turtlebot is facing. This will help the localization algorithm to find the correct starting pose of the robot in the map.

Now drive the turtlebot around the environment using the teleoperation node. You should see the robot's position in the map update in RViz.

3.2.3: Now you can use the navigation capabilities of the turtlebot. The turtlebot is able to plan paths to desired goals and move there autonomously.

To do this we need to start the navigation node on the turtlebot. Run the following command in addition to the localization node:

```
ros2 launch turtlebot4_navigation nav2.launch.py
```

3.2.4: Now you can use the 2D Nav Goal button in RViz to send navigation goals to the turtlebot. The interface works in the same way as the pose estimate button, but you can click on a point in the map where you want the turtlebot to go. The turtlebot will then plan a path to this goal and move there autonomously.

Task 3.3 Use the navigation: Now that everything is in place, make use of the navigation capability.

3.3.1: Write a node that invokes the `navigate_to_pose` action of the navigation node and moves your turtlebot alternating between two fixed positions on your map.

Here is a tutorial showcasing how to use the action using the command line interface (You can ignore the `/a300_0000` namespace they have in the example, we do not use it): `https://docs.clearpathrobotics.com/docs/ros/tutorials/navigation_demos/actions/`

Here is a tutorial showcasing how to use an action using Python: `https://docs.ros.org/en/humble/Tutorials/Intermediate/Writing-an-Action-Server-Client/Py.html#writing-an-action-client`

3.3.2: Add two folders to your package (on the same level as the package.xml file), one called `launch` and one called `config`. Move your map files into the `config` folder.

3.3.3: Create a XML launch file called `main.launch` in the `launch` folder that starts the localization, the navigation with your map and your node. To include the files in the installation process, replace the following line in your packages `setup.py` file:

```
    data_files=[
        (
            "share/ament_index/resource_index/packages",
            ["resource/" + package_name]
        ),
        (
            "share/" + package_name,
            ["package.xml"]
        ),
        (
            "share/" + package_name + "/config",
            ["tams_floor.yaml", "tams_floor.pgm"]  # Add your map files here
        ),
        (
            "share/" + package_name + "/launch",
            ["launch/main.launch"])   # Add your launch file here
    ],
```

This way the build tool knows what files to copy or link into the installation directory.

You can reference the build directory in the launch file using the `$(find-pkg-share <YOUR_PACKAGE>)` function. Just insert it into the path where you want to use it, e.g.:

```
<arg name="map" value="$(find-pkg-share <YOUR_PACKAGE>)/config/tams_floor.yaml" />
```

Remember to replace `<YOUR_PACKAGE>` with the name of your package.

3.3.4: After building / deploying your package, you can run your launch file using the following command:

```
ros2 launch <YOUR_PACKAGE> main.launch
```

This should start the localization, the navigation and your node, resulting in the turtlebot moving between the two fixed positions you specified in your node.