Home Service Robot

The goal of this project was to design a robot's environment in gazebo and program the home-service robot that will map its environment and autonomously navigate to pre-specified pickup and drop-off locations. For this one needed to:

- Design robot's environment with the Building Editor in Gazebo.
- Teleoperate the robot and manually test SLAM.
- Use the ROS navigation stack and manually command the robot using the 2D Nav Goal arrow in rviz to move to 2 different desired positions and orientations.
- Write a pick_objects node that commands the robot to move to the desired pickup and drop off zones.
- Write an add_markers node that subscribes to the robot odometry and publishes pick-up and drop-off markers to rviz.
- modify pick_objects node and add_markers node to establish communication between them, to complete desired home service robot implementation

Official ROS packages used:

1. gmapping: With the **gmapping_demo.launch** file, you can easily perform SLAM and build a map of the environment with a robot equipped with laser range finder sensors or RGB-D cameras.

2. turtlebot_teleop: With the **keyboard_teleop.launch** file, a robot can be manually controlled using keyboard commands.

3. turtlebot_rviz_launchers: With the **view_navigation.launch** file, a preconfigured rviz workspace is loaded, automatically launching the robot model, trajectories, and map.

4. turtlebot_gazebo: With the **turtlebot_world.launch** can deploy a turtle bot in a gazebo environment by linking the world file to it.

SLAM Testing: A shell script test_slam.sh is used to deploy a turtle bot inside your environment, control it with keyboard commands, interface it with a SLAM package, and visualize the map in rviz.

Localization and Navigation Testing:

For localization testing, a turtle bot is deployed in the world environment with a specific pose, is localized using amcl and observed in a map in rviz. The 2D Nav tab in rviz is used to manually point out to two different goals, one at a time, and direct the robot to reach them and orient itself with respect to them.

Packages created:

1. Navigation Goal Node (pick-objects): To test the robot's capability to reach multiple goals, as specified by the program (and not manually), a pick_objects package and specifically pick_objects_test.cpp function is created.

The algorithm used: This node communicates with ROS navigation stack and autonomously send successive goals for the robot to reach. As mentioned earlier, the ROS navigation stack creates a path for the robot based on Dijkstra's algorithm, a variant of the Uniform Cost Search algorithm, while avoiding obstacles on its path.

2. Virtual Objects Node (add-markers): To model a virtual object with markers in rviz, an add_markers package and specifically add_markers_test.cpp function is created.

   Algorithm:

- Publish the marker at the pickup zone
- Pause 5 seconds
- Hide the marker
- Pause 5 seconds
- Publish the marker at the drop off zone

Home-Service-Robot package

To simulate a full home service robot capable of navigating to pick up and deliver virtual objects, communication is established between the add_markers and pick_objects nodes via add_markers node subscribe to your robot odometry and keep track of your robot pose. For this purpose modified versions of the previous test, codes were created respectively pick_objects.cpp and add_markers.cpp. The entire package can be launched using a shell script.

The robot in this project complete these jobs successfully:

- Initially show the marker at the pickup zone
- Hide the marker once your robot reaches the pickup zone
- Wait 5 seconds to simulate a pickup
- Show the marker at the drop off zone once your robot reaches it.