# Home service turtlebot

Jung, Myoungki

**Abstract**—Building a robot able to mapping the surroundings and navigate through the environment based on the map and perform actions is non-tribual task. This project is to show the complete methodology to build one based on turtlebot.

**Index Terms**—IEEEtran, Udacity, ROS, Turtlebot.

## 1 Rubric Match

Majority of scripts methodology and scripts I used for this project is recorded sequentially on the `README.md` of this repository. This report file is to guide the location of the project requirement files and explains the functions of these files.

### 1.1 Basic Requirements

The ROS modules used in the project are included as `git submodules` in this repository. The root of the repository is to `catkin_ws/src/`. My own C++ ROS node based on the lesson 8,9,10,11 are included in directories `/wall_follower`, `pick_objects`, `add_markers` respectively.

### 1.2 Environment

The test world for slam and node writing is `World/WorldN/WorldN.world`. `World/WindMill/WindMill.world` was tested for the final project test.

### 1.3 Robot Model

It is possible to use my previous racerbot with D435 sensor however, it delays to troubleshoot errors without support available online or slack community. Therefore The default turtlebot was used throughout the project.

### 1.4 Mapping

Manual SLAM testing wasd done by the script `ShellScripts/test_slam.sh`. Mapping attempts using Gmapping withdefault parameter was almost impossible to build a clean and accurate map. The tuned custom launch file is `wall_follower/launch/gmapping.launch` shown on the listing 1 which is based on . The automated mapping script `ShellScripts/wall_follower.sh`.

```
      <launch>

  <arg name="scan_topic" default="/scan" />

  <node pkg="gmapping" type="slam_gmapping" name="
    slam_gmapping" output="screen">

    <param name="odom_frame" value="odom"/>
    <param name="base_frame" value="base_link"/>
    <param name="map_frame" value="map"/>

    <!-- Process 1 out of every this many scans (set
      it to a higher number to skip more scans) -->
    <param name="throttle_scans" value="1"/>

    <param name="map_update_interval" value="2.0"/>
    <!-- default: 5.0 -->

    <!-- The maximum usable range of the laser. A
    beam is cropped to this value. -->
    <param name="maxUrange" value="20.0"/>

    <!-- The maximum range of the sensor. If regions
     with no obstacl:quj;es within the range of the
    sensor should appear as free space in the map,
    set maxUrange < maximum range of the real sensor
     <= maxRange -->
    <param name="maxRange" value="25.0"/>

    <param name="sigma" value="0.05"/>
    <param name="kernelSize" value="1"/>
    <param name="lstep" value="0.05"/>
    <param name="astep" value="0.05"/>
    <param name="iterations" value="5"/>
    <param name="lsigma" value="0.075"/>
    <param name="ogain" value="3.0"/>
    <param name="minimumScore" value="800.0"/>
    <!-- Number of beams to skip in each scan. -->
    <param name="lskip" value="0"/>

    <param name="srr" value="0.01"/>
    <param name="srt" value="0.02"/>
    <param name="str" value="0.01"/>
    <param name="stt" value="0.02"/>

    <!-- Process a scan each time the robot
    translates this far -->
    <param name="linearUpdate" value="0.1"/>

    <!-- Process a scan each time the robot rotates
    this far -->
    <param name="angularUpdate" value="0.05"/>

    <param name="temporalUpdate" value="-1.0"/>
    <param name="resampleThreshold" value="0.5"/>

    <!-- Number of particles in the filter. default
    30       -->
    <param name="particles" value="50"/>

<!-- Initial map size -->
    <param name="xmin" value="-10.0"/>
    <param name="ymin" value="-10.0"/>
    <param name="xmax" value="10.0"/>
    <param name="ymax" value="10.0"/>

    <!-- Processing parameters (resolution of the
    map) -->
```

```
    <param name="delta" value="0.05"/>

    <param name="llsamplerange" value="0.01"/>
    <param name="llsamplestep" value="0.01"/>
    <param name="lasamplerange" value="0.005"/>
    <param name="lasamplestep" value="0.005"/>

    <remap from="scan" to="$(arg scan_topic)"/>
  </node>
</launch>
```

Listing 1: Python script to download dataset

The map files for testing are `World/WorldN/myMap.pgm` and `World/WorldN/myMap.yaml`. The saved final map files for this project are `World/WindMill/WindMill.yaml` and `World/WindMill/WindMill.pgm`.

### 1.5 Navigation

Manual navigation test was conducted with script `ShellScripts/test_navigation.sh` in Rviz. `pick_objects` contains a node sending goals on predefined time and conditions. By the ROS naming convention of the node, the node was named to `pick_objects_node`. The goal values were hardcoded in the pick value as these values are dependent on the map file . `ShellScripts/pick_objects.sh` file launches the node. The robot has to travel to the desired pickup zone, display a message that it reached its destination, wait 5 seconds, travel to the desired drop off zone, and display a message that it reached the drop off zone.

### 1.6 Markers

`add_markers` directory contains maker manager node, which shows and hides a marker in the scene based on robot's pose and goal setting publiser, `pick_objects_node`. `ShellScripts/add_markers.sh` script publish a marker to rviz. The initial `add_markers_node` published a marker should at the pickup zone. After 5 seconds it is hidden. Then after another 5 seconds it should appear at the drop off zone. The master branch is a modified `add_markers_node` serves the requirements for the home service bot. `add_markers/rviz/add_marker_cfg.rviz` was created to show the markers in Rviz. This Rviz configuration file is to be used for any tasks involved with this node.

### 1.7 HomeService

Subscription to odometry and `move_base/current_goal` topics enabled the `add_markers_node` to toggle marker shown when it is not owned or picked up by the robot or dropped by a user. The node hides the marker when it is picked up by the robot. After script `ShellScripts/home_service.sh` finishes, this behaviours can be observed by manually placing the goal in Rviz.

The node meets the requirements listed below.

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

## 2 Conclusion / Future work

The project was not a straight forward but left much room to study ROS especially to study building custom C++ nodes. The logic in wall follower node can be used to the very first Mars Rover project of the Udacity Robotics nano degree course. The project resembles the service oriented turtlebot [1]. This project can be extended to many different purpose. In addtion, porting to hardware instead of gazebo simulation environment can be challenging but rewarding future project.

## References

[1] A. Koubâa, M. F. Sriti, Y. Javed, M. Alajlan, B. Qureshi, F. Ellouze, and A. Mahmoud, "Turtlebot at Office: A Service-Oriented Software Architecture for Personal Assistant Robots Using ROS," in *Proceedings - 2016 International Conference on Autonomous Robot Systems and Competitions, ICARSC 2016*, 2016.