RBE 3002 – Final Project Report

Peerapat Luxsuwong

Tri Khuu

12/17/2015

Introduction

In the final project for this course, we were required to program the turtlebot to explore and generate a map of an unknown environment, which in this case, was an open space with some obstacles. Achieving this goal through any reasonable means was allowed, which meant that using the move_base navigation stack was an option, which was used in our ROS package's implementation for this project.

Methodology

This project was implemented across two machines: a remote laptop, and the laptop on the turtlebot. This allowed for remote access and control of the turtlebot via SSH and ROS.

Turtlebot Machine

Two commands had to be run on the turtlebot to allow for remote communication and control of the turtlebot:

1. rosrun turtlebot_bringup minimal.launch
2. rosrun turtlebot_navigation gmapping_demo.launch

Remote Machine

<u>ROS subscribers</u> had to be set up on the remote machine to allow for closed loop control of the turtlebot:

- odometry:
    - Subscribing to the /odom topic allowed the remote machine to update the turtlebot's known pose based off odometry information
- goal statuses:
    - Subscribing to the /move_base/status topic allowed the remote machine to keep track of the status of the turtlebot with regards to its goal poses

A <u>ROS publisher</u> also had to be set up on the remote machine:

- move base:
    - Publishing to the /move_base_simple/goal topic allowed the remote machine to control the turtlebot.

The <u>exploration & navigation algorithm</u> used by the turtlebot involved two main functions:

1. Detecting frontier fragments.
2. Navigating to a frontier fragment.

<u>Detecting frontier fragments:</u>

The frontier fragment detection function was called:

- On receipt of a new map.
- As a precursor to the exploring/navigation function.

The frontier fragment detection function can be broken down into the following steps. Sub-steps are also included for completeness and to detail how branch cases are dealt with:

1. A breadth-first search was run from turtlebot's current position to all fronter cells.
2. The frontier cells were merged into fragments based on their adjacency.
3. Frontier fragments that were too small for the turtlebot to navigate through were filtered out.
    a. The maximum width across a frontier fragment was used as the comparator.

<u>Navigating to a frontier fragment:</u>
The exploring/navigation function was called:

- On receipt of a new goal status that either indicated that it had been reached or was unreachable.

The exploring/navigation function can be broken down into the following steps. Sub-steps are also included for completeness and to detail how branch cases are dealt with:

1. The centroid of each frontier fragment was calculated.
    a. The x coordinates of all the cells in a fragment were averaged to find the x coordinate of that fragment's centroid.
    b. The y coordinates of all the cells in a fragment were averaged out to find the y coordinate of that fragment's centroid.
    c. If the resulting centroid was not empty, then the closest empty cell was found and returned instead.
2. The frontier fragments were sorted by some benchmark so as to gain a preferred frontier fragment's centroid.
    a. The benchmark used for each fragment in this case was:

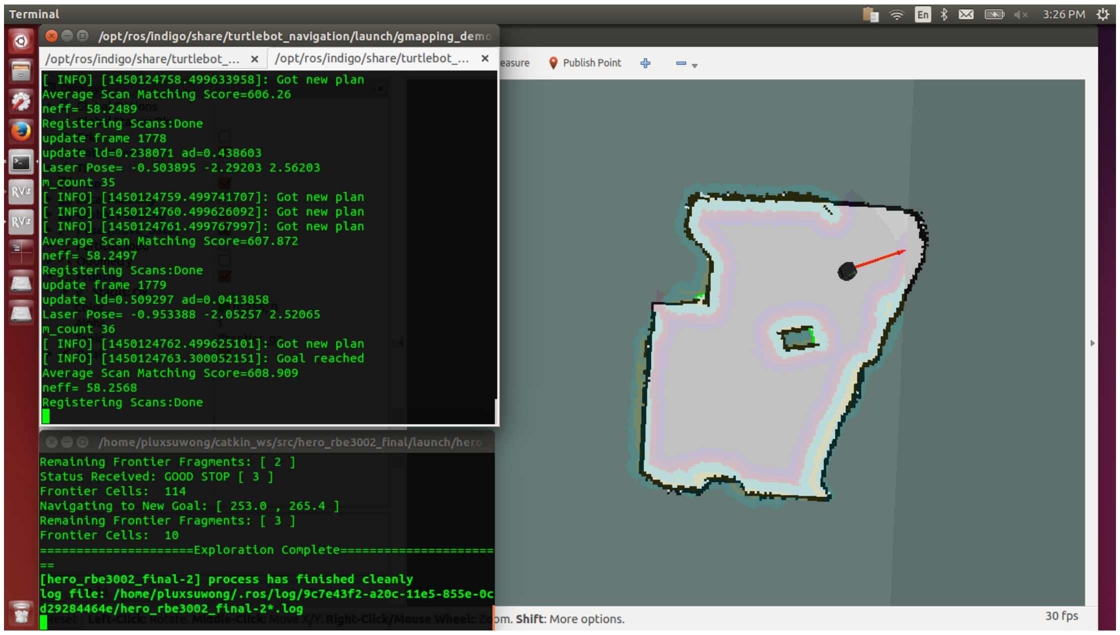
    $$\text{preference}_{\text{frag.}} = (\#\text{ of frag. cells})^2/(\text{dist. between frag. centroid and current position})$$

    *Note: A larger value corresponded to a higher preference.

3. The turtlebot navigated to the most preferred centroid.
    a. If the preferred centroid was unreachable (i.e. too close to an obstacle), another fragment's centroid was randomly selected instead.
    b. The turtlebot navigated to its goal using the move_base navigation stack, which provided path-planning and obstacle avoidance capabilities.
    c. Entropy was factored into the calculation of the waypoints (i.e. centroids in this case), so as to rely on randomness to force the turtlebot to proceed, should it by chance get stuck.

Results

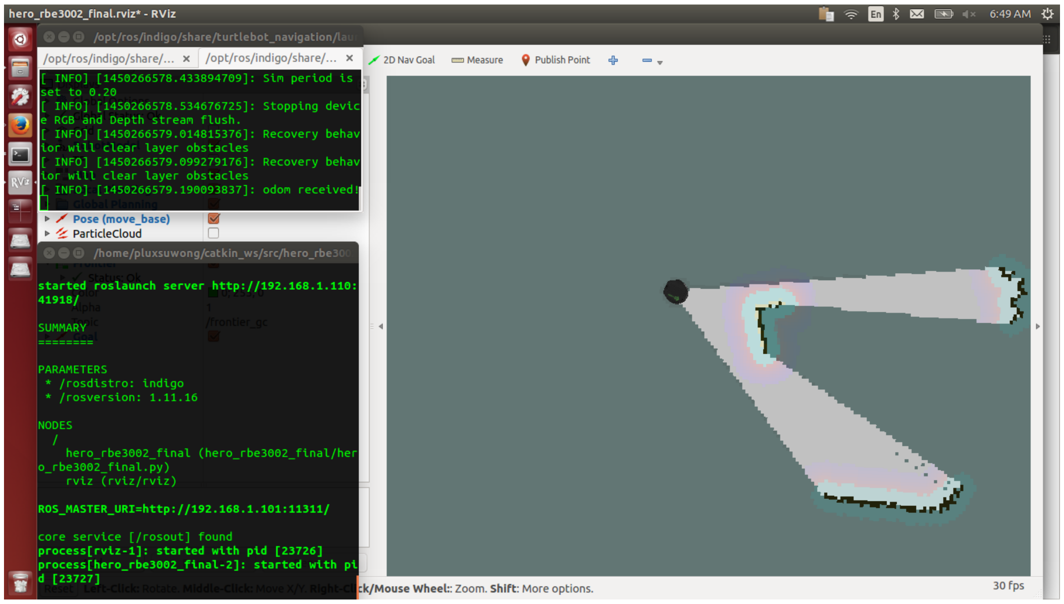
The overall findings were satisfactory. The turtlebot was able to successfully explore and map the provided enclosure. The following figures are some of the obtained results displayed in rviz.


Terminal
/opt/ros/indigo/share/turtlebot_navigation/launch/gmapping_demo
/opt/ros/indigo/share/turtlebot_...
Publish Point
3:26 PM
/home/pluxsuwong/catkin_ws/src/hero_rbe3002_final/launch/hero
30 fps

```
[ INFO] [1450124758.499633958]: Got new plan
Average Scan Matching Score=606.26
neff= 58.2489
Registering Scans:Done
update frame 1778
update ld=0.238071 ad=0.438603
Laser Pose= -0.503895 -2.29203 2.56203
m_count 35
[ INFO] [1450124759.499741707]: Got new plan
[ INFO] [1450124760.499626092]: Got new plan
[ INFO] [1450124761.499767997]: Got new plan
Average Scan Matching Score=607.872
neff= 58.2497
Registering Scans:Done
update frame 1779
update ld=0.509297 ad=0.0413858
Laser Pose= -0.953388 -2.05257 2.52065
m_count 36
[ INFO] [1450124762.499625101]: Got new plan
[ INFO] [1450124763.300052151]: Goal reached
Average Scan Matching Score=608.909
neff= 58.2568
Registering Scans:Done
```

```
Remaining Frontier Fragments: [ 2 ]
Status Received: GOOD STOP [ 3 ]
Frontier Cells:  114
Navigating to New Goal: [ 253.0 , 265.4 ]
Remaining Frontier Fragments: [ 3 ]
Frontier Cells:  10
=====================Exploration Complete=====================
==
[hero_rbe3002_final-2] process has finished cleanly
log file: /home/pluxsuwong/.ros/log/9c7e43f2-a20c-11e5-855e-0c
d29284464e/hero_rbe3002_final-2*.log
```

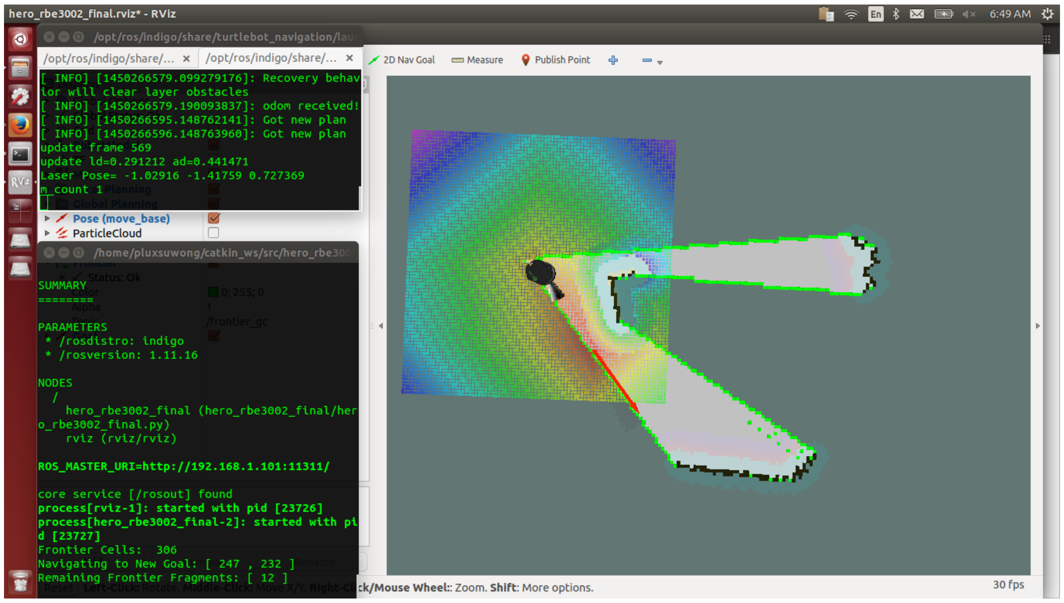
Figure 0: The cleanest map produced so far by the turtlebot. The enclosure in the following figures are the same one as this one, so this figure should make for a good benchmark of the quality of the results.


hero_rbe3002_final.rviz* - RViz
/opt/ros/indigo/share/turtlebot_navigation/lau
/opt/ros/indigo/share/...
2D Nav Goal
Measure
Publish Point
6:49 AM
Pose (move_base)
ParticleCloud
/home/pluxsuwong/catkin_ws/src/hero_rbe300
k/Mouse Wheel:: Zoom. Shift: More options.
30 fps

```
[ INFO] [1450266578.433894709]: Sim period is
set to 0.20
[ INFO] [1450266578.534676725]: Stopping devic
e RGB and Depth stream flush.
[ INFO] [1450266579.014815376]: Recovery behav
ior will clear layer obstacles
[ INFO] [1450266579.099279176]: Recovery behav
ior will clear layer obstacles
[ INFO] [1450266579.190093837]: odom received!
```

```
started roslaunch server http://192.168.1.110:
41918/

SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.16

NODES
  /
    hero_rbe3002_final (hero_rbe3002_final/her
o_rbe3002_final.py)
    rviz (rviz/rviz)

ROS_MASTER_URI=http://192.168.1.101:11311/

core service [/rosout] found
process[rviz-1]: started with pid [23726]
process[hero_rbe3002_final-2]: started with pi
d [23727]
```

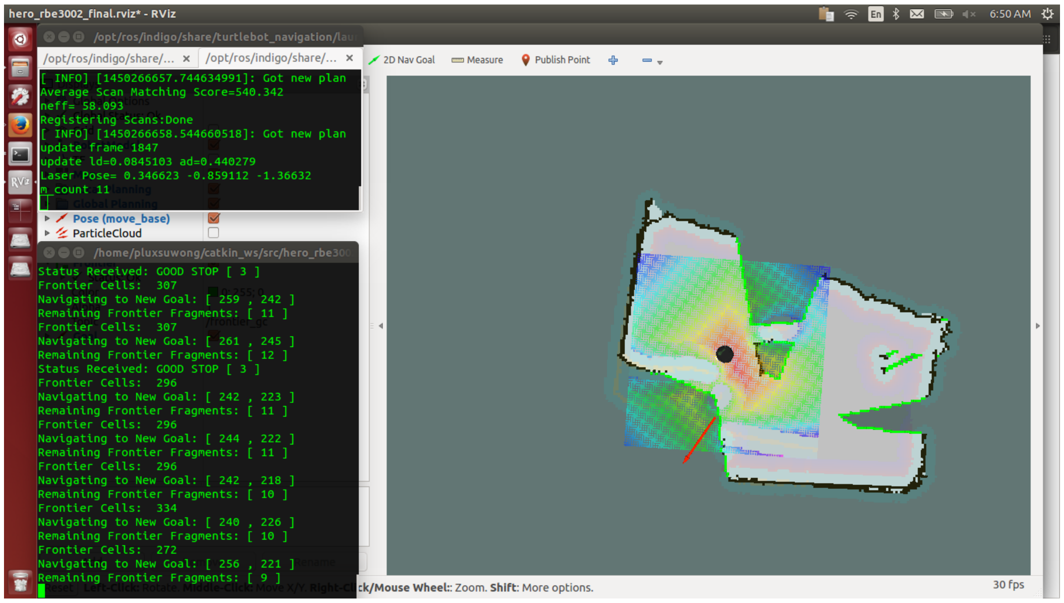
Figure 1: A fresh exploration/navigation session.



```
[ INFO] [1450266579.099279176]: Recovery behav
ior will clear layer obstacles
[ INFO] [1450266579.190093837]: odom received!
[ INFO] [1450266595.148762141]: Got new plan
[ INFO] [1450266596.148763960]: Got new plan
update frame 569
update ld=0.291212 ad=0.441471
Laser Pose= -1.02916 -1.41759 0.727369
m_count 1
```

```
SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.16

NODES
  /
    hero_rbe3002_final (hero_rbe3002_final/her
o_rbe3002_final.py)
    rviz (rviz/rviz)

ROS_MASTER_URI=http://192.168.1.101:11311/

core service [/rosout] found
process[rviz-1]: started with pid [23726]
process[hero_rbe3002_final-2]: started with pi
d [23727]
Frontier Cells:  306
Navigating to New Goal: [ 247 , 232 ]
Remaining Frontier Fragments: [ 12 ]
```

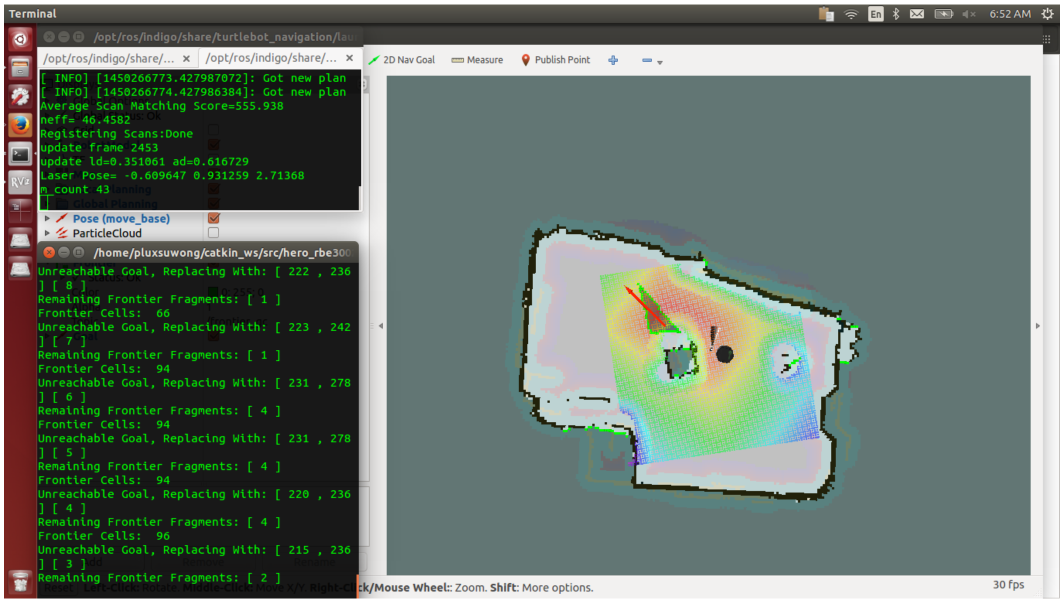
Figure 2: A continuation from Figure 1. The turtlebot had just received its first goal, [247, 232], and is travelling towards it. The colorful square is the turtlebot's local map, and the green lines mark the frontier fragments.



```
[ INFO] [1450266657.744634991]: Got new plan
Average Scan Matching Score=540.342
neff= 58.093
Registering Scans:Done
[ INFO] [1450266658.544660518]: Got new plan
update frame 1847
update ld=0.0845103 ad=0.440279
Laser Pose= 0.346623 -0.859112 -1.36632
m_count 11
```

```
Status Received: GOOD STOP [ 3 ]
Frontier Cells:  307
Navigating to New Goal: [ 259 , 242 ]
Remaining Frontier Fragments: [ 11 ]
Frontier Cells:  307
Navigating to New Goal: [ 261 , 245 ]
Remaining Frontier Fragments: [ 12 ]
Status Received: GOOD STOP [ 3 ]
Frontier Cells:  296
Navigating to New Goal: [ 242 , 223 ]
Remaining Frontier Fragments: [ 11 ]
Frontier Cells:  296
Navigating to New Goal: [ 244 , 222 ]
Remaining Frontier Fragments: [ 11 ]
Frontier Cells:  296
Navigating to New Goal: [ 242 , 218 ]
Remaining Frontier Fragments: [ 10 ]
Frontier Cells:  334
Navigating to New Goal: [ 240 , 226 ]
Remaining Frontier Fragments: [ 10 ]
Frontier Cells:  272
Navigating to New Goal: [ 256 , 221 ]
Remaining Frontier Fragments: [ 9 ]
```

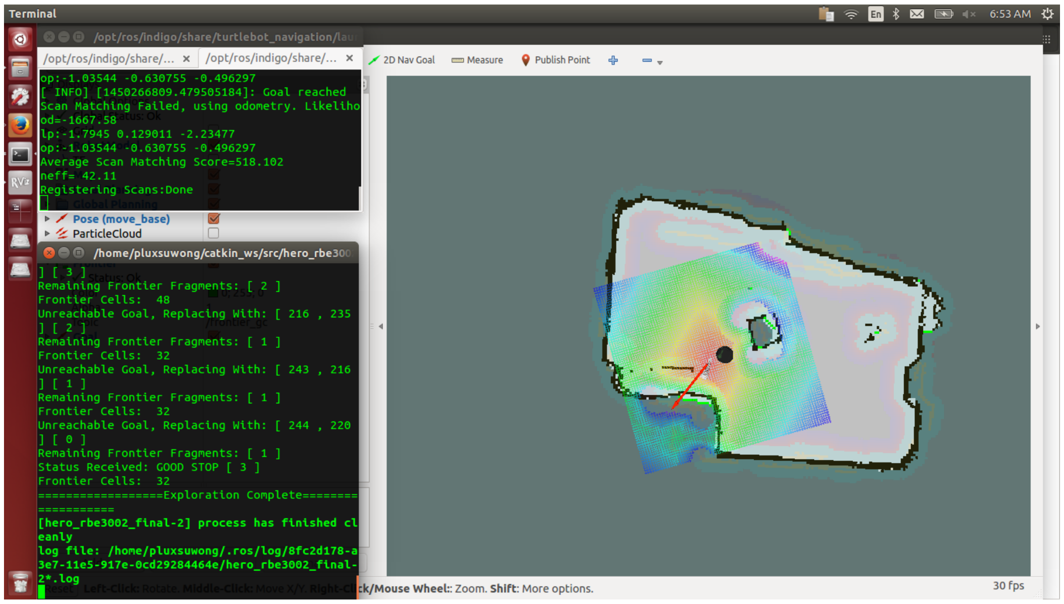
Figure 3: A continuation from Figure 2. After determining and navigating to several more goals, a much larger portion of the enclosure was mapped out by the turtlebot. Another stationary turtlebot remains mapped on the top-right corner of the map


Terminal
/opt/ros/indigo/share/turtlebot_navigation/lau
/opt/ros/indigo/share/...
/opt/ros/indigo/share/...
Pose (move_base)
ParticleCloud
/home/pluxsuwong/catkin_ws/src/hero_rbe300
2D Nav Goal
Measure
Publish Point
6:52 AM
30 fps

```
[ INFO] [1450266773.427987072]: Got new plan
[ INFO] [1450266774.427986384]: Got new plan
Average Scan Matching Score=555.938
neff= 46.4582
Registering Scans:Done
update frame 2453
update ld=0.351061 ad=0.616729
Laser Pose= -0.609647 0.931259 2.71368
m_count 43
```

```
Unreachable Goal, Replacing With: [ 222 , 236
] [ 8 ]
Remaining Frontier Fragments: [ 1 ]
Frontier Cells:  66
Unreachable Goal, Replacing With: [ 223 , 242
] [ 7 ]
Remaining Frontier Fragments: [ 1 ]
Frontier Cells:  94
Unreachable Goal, Replacing With: [ 231 , 278
] [ 6 ]
Remaining Frontier Fragments: [ 4 ]
Frontier Cells:  94
Unreachable Goal, Replacing With: [ 231 , 278
] [ 5 ]
Remaining Frontier Fragments: [ 4 ]
Frontier Cells:  94
Unreachable Goal, Replacing With: [ 220 , 236
] [ 4 ]
Remaining Frontier Fragments: [ 4 ]
Frontier Cells:  96
Unreachable Goal, Replacing With: [ 215 , 236
] [ 3 ]
Remaining Frontier Fragments: [ 2 ]
```

Figure 4: A continuation from Figure 3. Having mapped out the majority of the enclosure, the turtlebot is now focused on trying to navigate to an unreachable goal, which is in an unfortunate spot, which the turtlebot's BFS was still able to get to.


Terminal
/opt/ros/indigo/share/turtlebot_navigation/lau
/opt/ros/indigo/share/...
/opt/ros/indigo/share/...
Pose (move_base)
ParticleCloud
/home/pluxsuwong/catkin_ws/src/hero_rbe300
2D Nav Goal
Measure
Publish Point
6:53 AM
30 fps

```
op:-1.03544 -0.630755 -0.496297
[ INFO] [1450266809.479505184]: Goal reached
Scan Matching Failed, using odometry. Likeliho
od=-1667.58
lp:-1.7945 0.129011 -2.23477
op:-1.03544 -0.630755 -0.496297
Average Scan Matching Score=518.102
neff= 42.11
Registering Scans:Done
```

```
] [ 3 ]
Remaining Frontier Fragments: [ 2 ]
Frontier Cells:  48
Unreachable Goal, Replacing With: [ 216 , 235
] [ 2 ]
Remaining Frontier Fragments: [ 1 ]
Frontier Cells:  32
Unreachable Goal, Replacing With: [ 243 , 216
] [ 1 ]
Remaining Frontier Fragments: [ 1 ]
Frontier Cells:  32
Unreachable Goal, Replacing With: [ 244 , 220
] [ 0 ]
Remaining Frontier Fragments: [ 1 ]
Status Received: GOOD STOP [ 3 ]
Frontier Cells:  32
==================Exploration Complete========
===========
[hero_rbe3002_final-2] process has finished cl
eanly
log file: /home/pluxsuwong/.ros/log/8fc2d178-a
3e7-11e5-917e-0cd29284464e/hero_rbe3002_final-
2*.log
```

Figure 5: A continuation from Figure 4. The turtlebot, after relocalizing itself a couple of times, managed to affect its global costmap enough to generate additional frontier fragments. After navigating to one of them, the turtlebot deemed the map sufficiently mapped out, and terminated the process.

Discussion

As the final project to the final course in the Unified Robotics sequence at WPI, this project has two primary goals. The most outstanding one is to ensure that the students involved understand and are able to implement concepts involved in SLAM, to make the turtlebot explore and map out an unknown environment. The less obvious objective of this project, however, was to certify that the students are also able to utilize available resources and frameworks to achieve their own goals, as robotics in both industry and research is often used to solve complex problems. Creating a solution from scratch, which may have already been implemented, may not be the optimal method to dealing with such a scenario. In this project, both objectives were achieved.

The move_base navigation stack, a professionally developed open source ROS suite, was used as a foundation to add the desired functionality to the turtlebot. Though ros packages providing similar functions to those of the move_base stack were developed in previous weeks of the course, the move_base stack was a well documented alternative that incorporated the same packages, and even more, as a cohesive whole than could have been achieved otherwise. As such, the secondly stated objective has been fulfilled as such.

It may be argued that the first objective might not have been fulfilled at all, due to the move_base stack providing and presenting much of the desired SLAM functionality as a black box, able to be interfaced with through functions. However, having personally interfaced with the move_base stack in the package we developed, and extensively tested the package with the turtlebot, we insist that it is not possible to successfully use such a complex piece of software without gaining some insight to why we received the results we did.

For instance, one of the requirements of a successful demonstration of autonomous SLAM on the turtlebot would be for the turtlebot to know when it has obtained a satisfactory mapping of its environment. This was achieved through the calculation and analysis of frontier fragments, which are indicative of boundaries on the turtlebot's knowledge. Knowledge of when to pursue the exploration of a fragment versus when to not are a necessity when dealing with systems incorporating SLAM, as though SLAM is indeed a powerful tool, errors involved in the process build up over time and do not guarantee the same results as the ones that are expected. Having considered and successfully planned ahead for these errors (i.e. IndexError exceptions for cells that are out of bounds) is another indication of mastery of the used material.

Being able to interpret and explain our results is also indicative of the satisfactory fulfillment of this project's criteria. In Figure 3, for instance, the light blue circle that can be seen in the turtlebot's local costmap near the turtlebot is another turtlebot, which was also currently exploring the enclosure. The imperfections in the map's edges in Figures 4 and 5 can also be explained: odometry. As the turtlebot tries to localize itself amongst its immediate surroundings and its stored version of the global map, it tries to determine where it is. Once it has done so, it updates its global map with new sensor data. An instance of where this occurred can be seen in Figure 5, where the turtlebot's odometry was slightly off, leading to slight transformations in different parts of the global map. Though this may appear to be detrimental in the short-run, due to the move_base stack using amcl (adaptive monte-carlo localization) as its localization method, the global map will eventually come to reflect the real world, should the turtlebot be allowed to continuously patrol the map to gather data.

There were two main problems that arose during the development of the package. One involved the method used to determine frontier cells. The dispute was between iterating throughout the entire costmap provided by the turtlebot and using a breadth-first search. Having developed and tested both, we arrived at the conclusion that though the BFS solution did take more time to program and debug, the vastly faster performance and lack of unreachable frontier cells was worth the time and effort.

The second problem was with regards to how to go about selecting waypoints. One outstanding solution was to implement our own A* functionality to allow for the selection of more conservative waypoints (to prevent unreachable waypoints from being selected). However, we came up with a simpler, albeit less elegant solution: the most preferred fragment centroid would be initially used. On the off-chance of its unreachability, another randomly selected centroid would be used, albeit to improve the chances of it being reachable, a slightly closer point to the turtlebot's position would be used.

At first, using the latter waypoint selection method, the turtlebot had trouble navigating to frontier fragments around a corner, as the centroids to these fragments would often end up too near a wall or obstacle, and hence unreachable space. To deal with this, instead of selecting the nearest, largest frontier fragment as the highest priority, the exact opposite was used: the farthest, smallest frontier fragment. Though counterintuitive, this implementation allowed for the turtlebot to passively explore the map as it would often navigate to a physically distant goal (which would often be in an almost straight line). This also drastically reduced the chance of there being unexplored spots around corners, which resolved the pending issue.

Conclusion

Developing a SLAM navigation ROS package for the turtlebot allowed for such theoretical concepts covered in class as odometry and localization to be applied in a real world scenario, while also developing our ability to incorporate existing modules into our project. While there is no single best answer to any of the complex problems that robotics attempts to tackle, there are many viable solutions, which may require testing and comparison to arrive at a conclusion, much like how we refined our waypoint selection method.

References

stackoverflow.com/questions/tagged/ros
docs.ros.org
wiki.ros.org