Project: Where am I

Jd Fiscus

1 Abstract

The second project of Udacity Robotics Nanodegree - Term 2 was to initially create a reference robot. Next is optimizing the parameters related to localization with the intention to have the robot move within a Gazebo maze, and ultimately find a predetermined end point.

The final goal is to create a new model by customizing the base and sensors. However, the intention is to reach a predetermined end point is the same.

2 Introduction

In the project the robot model doesn't know where it is in the maze, so it needs to create a local map of the environment utilizing senor feedback. By leveraging the map it can navigate to the predetermined end point.

For the local map to be created it has to lean on both ROS Navigation[1] and move_base[2] packages. These packages leverage the robot moving in relation to the global map which defines a path to continue moving. In this case global map was in Gazebo and provided to students.

The robots goal is reached only when it has reached its predetermined end point, no matter how long it takes to create the local map.

3 Background / Formation

When a robot is placed in an environment it has to acquire understanding of the location, so it can determine how to move throughout the environment to different points. This idea is more commonly known as localization[3] which is used in many different technologies.

The ROS localization package used in this project is called Adaptive Monte Carlo Localization (AMCL)[4]. This is a variation of Monte Carlo Localization (MCL)[5] another localization algorithm.

[1] "navigation - ROS Wiki - ROS.org." 10 Mar. 2017, http://wiki.ros.org/navigation. Accessed 26 May. 2018.
[2] "move_base - ROS Wiki - ROS.org." 26 Dec. 2016, http://wiki.ros.org/move_base. Accessed 26 May. 2018.
[3] "Localization - Wikipedia." https://en.wikipedia.org/wiki/Localization. Accessed 26 May. 2018.
[4] "amcl - ROS Wiki - ROS.org." http://wiki.ros.org/amcl. Accessed 26 May. 2018.

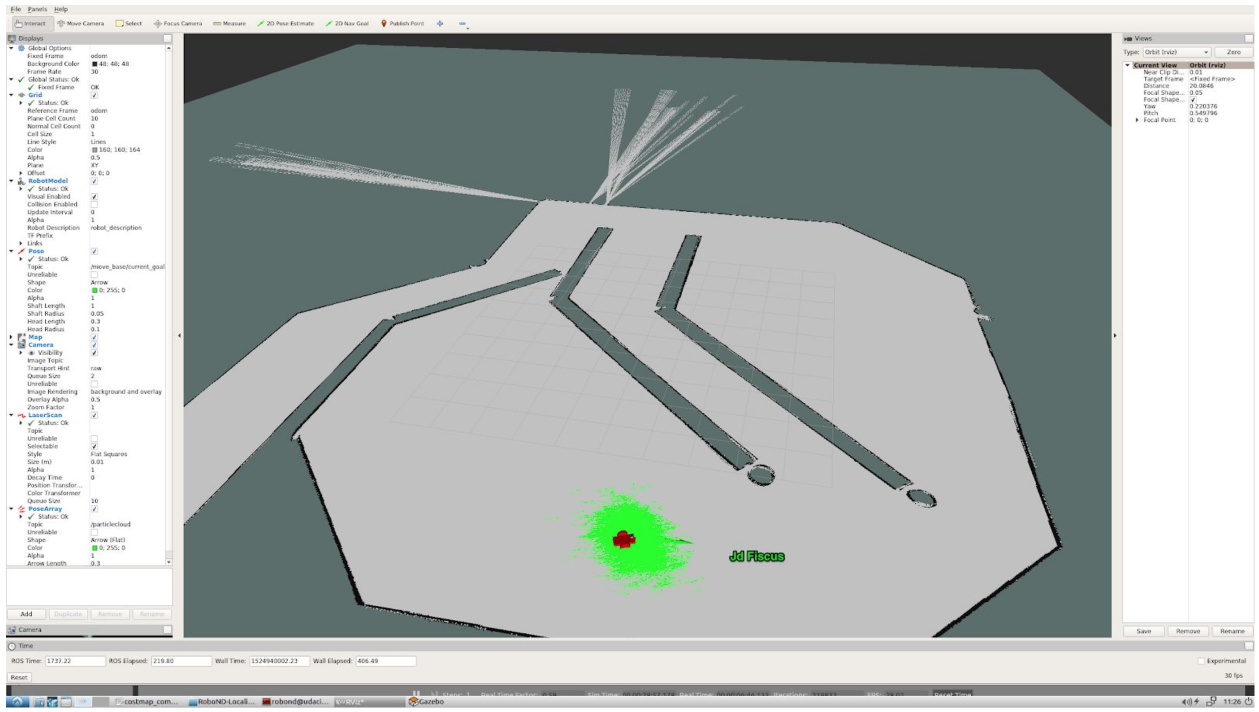
MCL can help localize a robot by using particle filters. It can use sensors to derive a particle which in this case is a camera and LIDAR attached to the robot.

4 Results

The first objective is to build and use the udacity_bot configured to move to a predetermined spot in the maze. Next was to modify that robots appearance and tune the parameters. In order to maintain the previous robot state, a copy was made and named iamficus_bot[6].

Here are the final results of each bot.

Udacity Bot



Global Options	
Fixed Frame	odom
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
✓ Status: Ok	
Reference Frame	odom
Plane Cell Count	10
Normal Cell Count	0
Cell Size	1
Line Style	Lines
Color	▨ 160; 160; 164
Alpha	0.5
Plane	XY
Offset	0; 0; 0
RobotModel	✓
✓ Status: Ok	
Visual Enabled	✓
Collision Enabled	
Update Interval	0
Alpha	1
Robot Description	robot_description
TF Prefix	
Links	
Pose	✓
✓ Status: Ok	
Topic	/move_base/current_goal
Unreliable	
Shape	Arrow
Color	🟩 0; 255; 0
Alpha	1
Shaft Length	1
Shaft Radius	0.05
Head Length	0.3
Head Radius	0.1
Map	✓
Camera	✓
Visibility	✓
Image Topic	
Transport Hint	raw
Queue Size	2
Unreliable	
Image Rendering	background and overlay
Overlay Alpha	0.5
Zoom Factor	1
LaserScan	✓
✓ Status: Ok	
Topic	
Unreliable	
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transfor...	
Color Transformer	
Queue Size	10
PoseArray	✓
✓ Status: Ok	
Topic	/particlecloud
Unreliable	
Shape	Arrow (Flat)
Color	🟩 0; 255; 0
Alpha	1
Arrow Length	0.3

Type: Orbit (rviz) Zero

Current View	Orbit (rviz)
Near Clip Di...	0.01
Target Frame	<Fixed Frame>
Distance	20.0846
Focal Shape...	0.05
Focal Shape...	✓
Yaw	0.220376
Pitch	0.549796
Focal Point	0; 0; 0

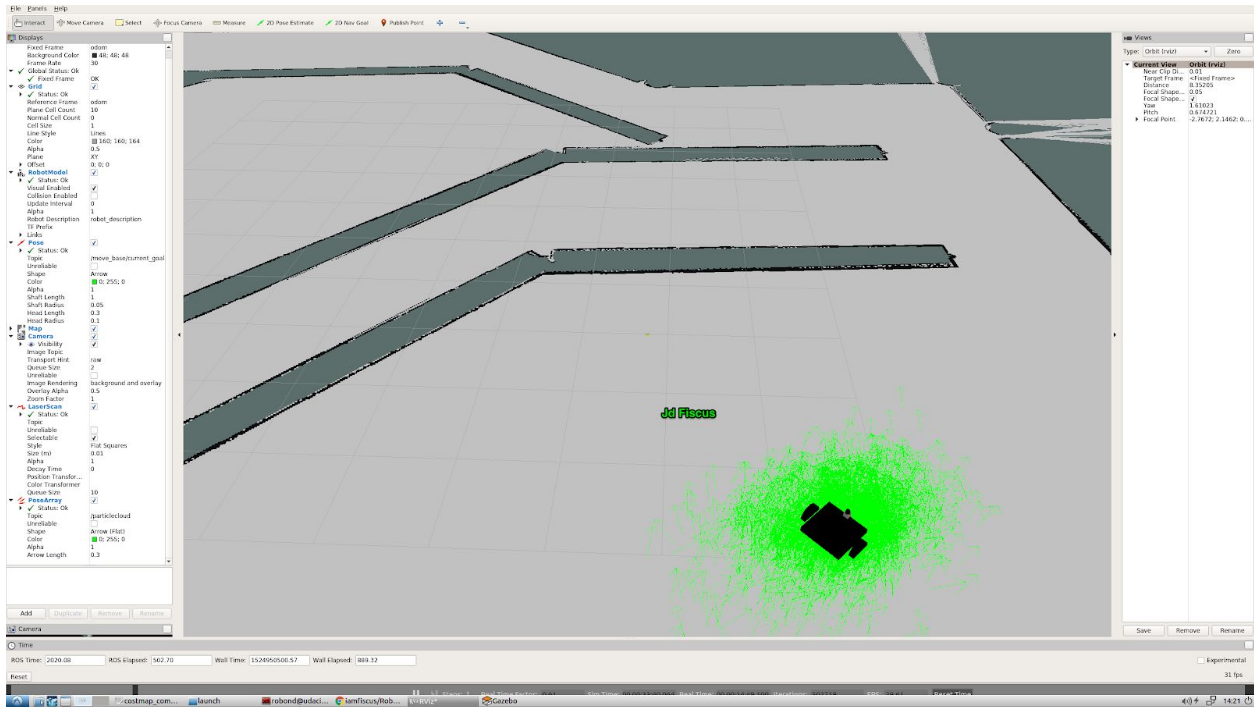
ROS Time: 1737.22 ROS Elapsed: 219.80 Wall Time: 1524940002.23 Wall Elapsed: 406.49

---

[5] "Monte Carlo localization - Wikipedia." https://en.wikipedia.org/wiki/Monte_Carlo_localization. Accessed 26 May. 2018.

[6] "iamfiscus (Jd Fiscus) · GitHub." https://github.com/iamfiscus/RoboND-Localization-Project. Accessed 27 May. 2018.

Iamfiscus Bot



```
Displays
Fixed Frame            odom
Background Color       48; 48; 48
Frame Rate             30
Global Status: Ok
  Fixed Frame          OK
Grid                   ✓
  Status: Ok
  Reference Frame      odom
  Plane Cell Count     10
  Normal Cell Count    0
  Cell Size            1
  Line Style           Lines
  Color                160; 160; 164
  Alpha                0.5
  Plane                XY
  Offset               0; 0; 0
RobotModel             ✓
  Status: Ok
  Visual Enabled       ✓
  Collision Enabled
  Update Interval      0
  Alpha                1
  Robot Description    robot_description
  TF Prefix
  Links
Pose                   ✓
  Status: Ok
  Topic                /move_base/current_goal
  Unreliable
  Shape                Arrow
  Color                0; 255; 0
  Alpha                1
  Shaft Length         1
  Shaft Radius         0.05
  Head Length          0.3
  Head Radius          0.1
Map                    ✓
Camera                 ✓
  Visibility           ✓
  Image Topic
  Transport Hint       raw
  Queue Size           2
  Unreliable
  Image Rendering      background and overlay
  Overlay Alpha        0.5
  Zoom Factor          1
LaserScan              ✓
  Status: Ok
  Topic
  Unreliable
  Selectable           ✓
  Style                Flat Squares
  Size (m)             0.01
  Alpha                1
  Decay Time           0
  Position Transfor...
  Color Transformer
  Queue Size           10
PoseArray              ✓
  Status: Ok
  Topic                /particlecloud
  Unreliable
  Shape                Arrow (Flat)
  Color                0; 255; 0
  Alpha                1
  Arrow Length         0.3

Views
Type: Orbit (rviz)     Zero
Current View           Orbit (rviz)
  Near Clip Di...      0.01
  Target Frame         <Fixed Frame>
  Distance             8.35205
  Focal Shape...       0.05
  Focal Shape...       ✓
  Yaw                  1.61023
  Pitch                0.674721
  Focal Point          -2.7672; 2.1462; 0....

Time
ROS Time: 2020.08   ROS Elapsed: 502.70   Wall Time: 1524950500.57   Wall Elapsed: 889.32
```

5 Model Configuration

In the case of iamfiscus_bot the first was to change the robots appearance which was to make it square and to add the color black to it. For the most part it is based on the same AMCL configuration as the udacity_bot.

The min and max particles were set to the default range of 25 and 200. Although other ranges were tested having max_particles even go up to 1000 there wasn't much improvement to the robot to localize itself, and the CPU usage would have spiked. Also the laser scan sensor was left as it was in the udacity_bot.

Costmap Common Params

Since the body of the robot was changed to a square base the first change was to the robot_radius equal to 0.5.

Next was to the change obstacle_range to 2, raytrace_range to 5, and inflation_radius to 0.75 which gave enough space for the robot to navigate in the cost map.

The transform_tolerance was set to 0.2, which was the same used in the udacity_bot.

Local Cost Map Params

No changes were made to the original settings from the udacity_bot.

Global Cost Map Params
No changes were made to the original settings from the udacity_bot.

Base Local Planner Params
No changes were made to the original settings from the udacity_bot.

6 Discussion

Every run the robot was able to complete the objective of finding the predetermined point. However, quite often the robot might head in a longer direction in order to complete the objective. This resulted in a longer timeframe for the objective to be completed.

It was very interesting to see how the robot can so quickly localize itself using ACML and then begin moving with the particle filters helping to adjust its path. However it requires more tuning to optimize its performance.

When talking about the kidnapped robot problem[7] which means a robot is placed in a arbitrary location ACML quickly localize itself using ACML and then begin moving with the particle filters that adapt helping to adjust its path. Therefore, it doesn't rely on the environment to provide information on where it's located.

The environments that ACML would work great in are things that are constantly changing. For instance a hospital which has equipment, practitioners, and patients moving. Also a residence or commercial space where obstacles change regularly.

7 Future Work

When the robot seemed to take the longer path it created a much larger cost map which seems it would have an impact on performance. Also, when changing the max_particles to 1000 there was a significant cost to CPU, while not having an impact on ACML. Although the defaults worked, fine tuning the range for the particles would increase the ability adjust it's path for better mapping.

It definitely seemed that the square robot performed better with turning specifically when hitting an obstruction like a wall. While the rectangle seemed to get stuck in place, the square was able to rotate in a circular fashion.

[7] "Kidnapped robot problem - Wikipedia." https://en.wikipedia.org/wiki/Kidnapped_robot_problem. Accessed 28 May. 2018.

The sensor which makes the most sense to add would be a LIDAR which could go on each side of the robot. However, depending on cost and compute power a 360 LIDAR would be optimal in creating a detailed map of the environment.