Custom Localization Robots

George Erol Fouche

Abstract—This project is about creating and testing 2 robots in a Robot Operating System(ROS) simulation environment. The ***UdacityBot*** and ***FouliexBot*** use Adaptive Monte Carlo Localization(AMCL) combined with a navigation stack to navigate a maze and reach a predefined destination position.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Localization.

1 Introduction

Localization is the challenge of determining a robot's pose in a mapped environment. This is done by implementing a probabilistic algorithm to filter noisy sensor measurements and track the robot's position and orientation. There's more than one way to localize a robot. There's 4 very popular localization algorithms.

The **Extended Kalman Filter** localization which is the most common Gaussian filter that helps in estimating the state of non-linear models. Another one is the **Markov Localization**, which is a base filter localization algorithm. Markov maintains a probability distribution over the set of all possible position and orientation the robot might be located at. In addition, there's the **Grid Localization** which refers to histogram filter since it's capable of estimating the robot's pose using grids.There's the **Monte Carlo Localization**, also known as particle filter because it estimates the robot's pose using particles.At last, there's the **Adaptive Monte Carlo Localization**(AMCL) which dynamically adjusts the number of particles over a period of time, as a robot navigates around in a map.

1.1 Challenges with Localization

There are 3 different types of localization challenges which are:

1) Local Localization or Position Tracking
2) Global Localization
3) Kidnapped Robotics

1.1.1 Position Tracking

The easiest problem with localization is called Position Tracking also known as local localization. In this problem, the robot knows its initial pose and localization challenge entails estimating the robot's pose as it moves out on the environment. This problem is not a trivial as one might think because there is always some uncertainty in robot motion. However, the uncertainty is limited to regions surround the robot.

1.1.2 Global Localization

A more complicated localization challenge is called Global Localization, in this case, the robot's initial pose is unknown and the robot must determine its pose relative to the ground truth map. The amount uncertainty in Global Localization is much greater than that in Position Tracking.

1.1.3 Kidnapped Robotics

The most challenging localization problem is the Kidnapped Robot problem. This problem is just like Global Localization except that the robot may be kidnapped at any time and moved to a new location on the map. For example, someone pick up the robot and but it at a difference place. This problem is not cover in this project.

In this project, 2 robots were developed and tested in a ROS simulation environment to navigated a maze with no Position tracking or Global Localization problem using the Adaptive Monte Localization (AMCL) algorithm.

2 Background

2.1 Extended Kalman Filters

The assumption under which Kalman Filter operates are that the motion and measurement functions are linear and that the state space can be represented by a unimodal Gaussian distribution. However, these assumptions are very limiting and would only works for very primitive robots. Most mobile robots will execute non-linear motions such as moving in a circle or following a curve. The Extended Kalman filter(EKF) address these limiting assumptions by linearizing a nonlinear motion or measurement function with multiple dimensions using multidimensional Taylor series

2.2 Adaptive Monte Localization

The Monte Carlo Localization algorithm is the most popular localization algorithms in robotics. Therefore, the Adaptive version of it is use in this project and deploy on both UdacityBot and FouliexBot to keep track of both robots poses.

Both robots are able to navigate inside a known map and collecting sensory information using range finder sensors. AMCL used these sensor measurements to keep track of both robot poses.

AMCL uses particles to localize the robots, one way to think of a particles is to think of virtual element that resembles the robot. Each particles has a position and orientation and represent a guess of where the robot might be located. These particles are re-sampled each time the robot moves and sense its environment.

2.3 Extended Kalman Filters vs Adaptive Monte Localization

With the Extended Kalman Filter localization algorithm one can estimate the pose of almost any robot with accurate on board sensors. However, AMCL presents many advantages over EKF. First, AMCL is easy to program compared to EKF. Second, AMCL represents non-Gaussian distributions and can approximate ant other practical important distribution. This mean that AMCL is unrestricted to linear Gaussian states-based assumption as is the case for EKF. This allows AMCL to model a much greater variety of environments especially since one can't really model the real world with Gaussian distributions. Third, in AMCL, one can control the computational memory and resolution of a solution by changing the number of particles distributed uniformly and randomly throughout the map. The differences between AMCL and EKF is summarized in table 1.

TABLE 1
Extended Kalman Filters vs Adaptive Monte Localization

	AMCL	EKF
Measurements	Raw measurements	Landmarks
Measurement Noise	Any	Gaussian
Posterior	Particles	Gaussian
Ease of Implementation	easy	not easy
Global localization	Yes	No
State Space	Multimodal Discrete	Unimodal Continuous

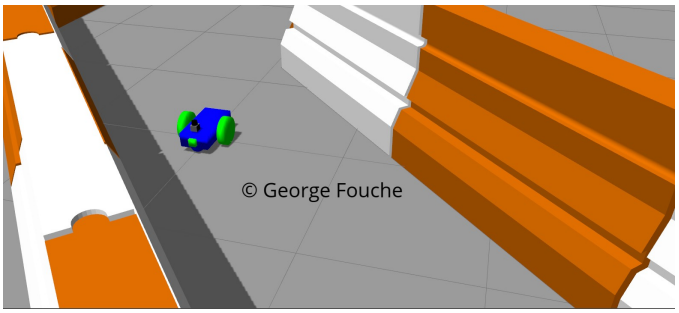

Fig. 1. UdacityBot in the Gazebo world Close Up

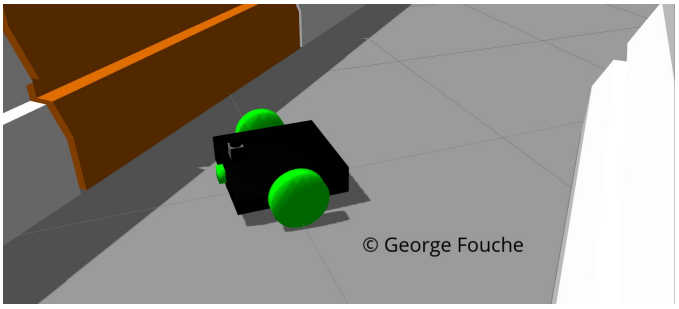

Fig. 2. FouliexBot Close UP

2.4 Model and Environmental Design

UdacitBot has a rectangular chassis with a camera ahead and a laser scanner at the top head of the chassis(Fig 1 and 4).FouliexBot has a square chassis with a camera ahead and a laser scanner at the top head of the chassis(Fig 2 and 3). The 2 URDF files defines the shape of the robots. The .gazebo file define the view and the .xacro provides the robot shape description.

2.4.1 Map

The ClearPath jackaml_race.yaml and jackal_race.pgm package are use to create the maps.

2.4.2 Meshes

Hokuyo scanner is the laser scanner use in this project. The hokuyo.dae mesh is use to render the laser projection.

2.4.3 Launch

The same three launch scripts with different naming are used for the UdacityBot and FouliexBot.
FouliexBot launch files:

1) amcl.launch
2) fouliex_world.launch
3) robot_description.launch

UdacityBot launch files:

1) amcl.launch
2) udacity_world.launch
3) robot_description.launch

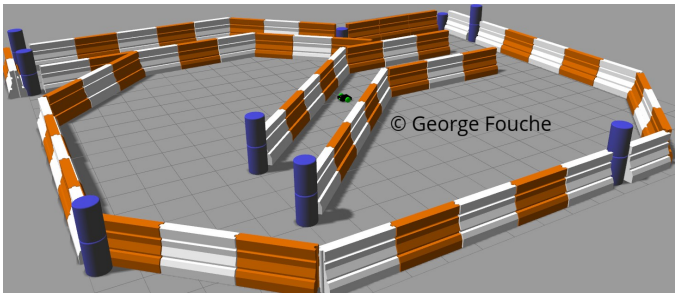

Fig. 3. FouliexBot in the Gazebo world

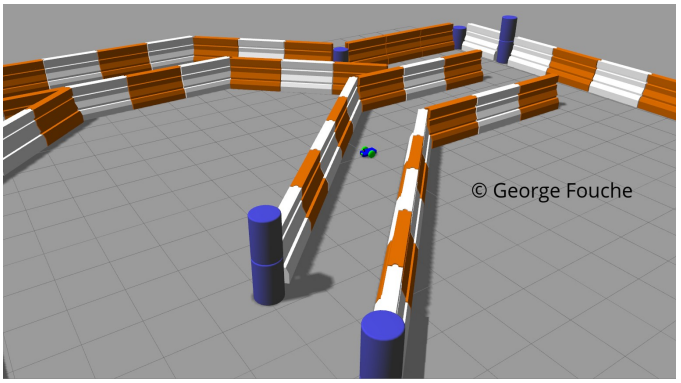

Fig. 4. UdacityBot in the Gazebo world

2.5 Model Configuration

2.6 AMCL Localization Configuration

UdacityBot and fouliexBot has different AMCL configuration.

2.6.1 Minimum and Maximum particles

As the AMCL dynamically adjusts its particles for every iteration, it expects a range of the number of particles as an input.This range is tuned based on the system specifications.A large range, with a high maximum might be too computationally extensive for a low-end system. For this both robot the minimum particles is set to 20 and the maximum particles is set to 200.

2.6.1.1 Min and Max particles parameters for both robots:

```
<param name="min_particles" value="20"/>
<param name="max_particles" value="200"/>
```

2.6.2 Transform Tolerance

The Transform Tolerance is the most important parameters to work with and to tune properly.It's also dependent of the system specifications. It helps decide the longevity of the transform being published for localization purposes. A good value should only be to account for any lags in the system.

2.6.2.1 Transform Tolerance parameter for UdacityBot:

```
<param name="transform_tolerance"
 ↪ value="0.2"/>
```

2.6.2.2 Transform Tolerance parameter for FouliexBot:

```
<param name="transform_tolerance"
 ↪ value="1.25"/>
```

2.6.3 Initial Pose

The initial pose is set to [0,0,-0.785] which for the x and y initial pose is set to 0 and the yaw initial pose is set to -0.785 for UdacityBot. For FouliexBot the values are [0,0,0]

2.6.3.1 Inital Pose paramters for UdacityBot:

```
<param name="initial_pose_x" value="0.0"/>
<param name="initial_pose_y" value="0.0"/>
<param name="initial_pose_a"
 ↪ value="-0.785"/>
```

2.6.3.2 Inital Pose paramters for UdacityBot:

```
<param name="initial_pose_x" value="0.0"/>
<param name="initial_pose_y" value="0.0"/>
<param name="initial_pose_a" value="0.0"/>
```

2.6.4 Laser

There are two different types of laser model to consider - the likelihood field and the beam. The likelihood field model is usually more computationally efficient and reliable for an environment such as this project environment therefore the laser model type for both robots is set to the likelihood field.

2.6.4.1 Laser parameters:

```
<remap from="scan"
 ↪ to="fouliex_bot/laser/scan"/>
<param name="laser_model_type"
 ↪ value="likelihood_field_prob"/>
```

```
<remap from="scan"
 ↪ to="udacity_bot/laser/scan"/>
<param name="laser_model_type"
 ↪ value="likelihood_field_prob"/>
```

2.6.5 Odometry

Since this project is using a differential drive mobile robot, it's best to use the diff-corrected type for both robots.

2.6.5.1 Odometry parameters:

```
<param name="odom_frame_id" value="odom"/>
<param name="odom_model_type"
 ↪ value="diff-corrected"/>
```

2.7 AMCL Localization Move Base

The Move Base package help navigate the robot to the goal position by creating or calculating a path from the initial position to the goal and the amcl package will localize the robot. The Move Base package consist of:

1) Local and Global Costmap parameters
2) Costmap Common parameters
3) Base Local Planner parameters

2.7.1 Local and Global Costmap

The Local Costmap relies on odom as a global frame since it updates as the robot moves forward. Since the costmap updates itself at specific intervals, and aims to cover a specific region around the robot, it requires it's own updating and publishing frequencies as well as dimensions for the costmap. The costmap performs map update cycles at the rate specified by the Update Frequency parameter. Each cycle, sensor data comes in, marking and clearing operations are performed in the underlying occupancy structure of the costmap, and this structure is projected into the costmap where the appropriate cost values are assigned. The Local Costmap update and publish frequency is set to 10 for UdacityBot and 3 for fouliexBot. The same update and publish frequency as the Local Costmap is set for the Global Costmap.

2.7.2 Costmap Common parameters

The Costmap Common consist of parameters related to the laser sensor such as:

1) The Obstacle Range
   - the obstacle range parameter determines the maximum range sensor reading that will result in an obstacle being put into the costmap. For UdacityBot, it set at 5.0 meters which means that the Both robots will only update its map with information about obstacles that are within 5.0 meters of the base. For fouliexBot it is set to 1.5.

2) The Raytrace Range

- The raytrace range parameter determines the range to which one can raytrace free-space given a sensor reading. Setting it to 8.0 meters means that the robot will attempt to clear out space in front of it up to 8.0 meters away given the sensor reading from the laser.For UdacityBot the raytrace is set to 8.0 and for FoubliexBot the raytrace is set to 4.0

3) The Inflation Radius

- The inflation radius should be set to the maximum distance from obstacles at which a cost should incurred. Setting the inflation radius at 0.55 meters means that the the robot will treat all paths that stay 0.55 meters or more, away from obstacles as having equal obstacle cost.For UdacityBot the inflation radius is set to 0.55 meters and for UdacityBot it is set to 0.65 meters.

2.7.3 Base Local Planner Parameters

The Move Base package creates and calculates a path or a trajectory to the goal position, and navigates the robot along that path. The set of parameters in Base Local Planner customize this particular behavior. The Base Local Planners parameters are:

TABLE 2
Base Local Planner Parameters for UdacityBot

Parameter	Value	Description
Sim time	1	It is the amount of time to forward-simulate trajectories in seconds.
Controller Frequency	10	It is the frequency at which the controller is called in Hz.
Pdist Scale	0.6	It is the weighting for how much the controller should stay close to the path it was given
Yaw Goal Tolerance	0.05	It help define how close the robot's pose can be to the goal position
XY Goal Tolerance	0.05	Same description as Yaw Goal Tolerance

Other parameters such are robot configuration parameters are located under the Base Local Planner file.

3 Result

UdacityBot was able to navigate to the destination path with no problem(Fig. 5 and Fig. 6). FouliexBot had a lot of difficulties but from time to time it will turn back after making the turn, not continuing to the destination path(Fig.7). The difference in mass and shape had a lot of effect on reaching the destination goal for FouliexBot. In average it took UdacityBot 10 mins to reach the goal and for FouliexBot up to 1 hr if it feels like it. Both robots were able to avoid collision with the barriers, therefore showing that localization worked but not navigation. As one can see it took FouliexBot almost 1 hour to reach the goal destination and from time to time it will stop and then turn around therefore not reaching the destination point.

TABLE 3
Base Local Planner Parameters for FouliexBot

Parameter	Value	Description
Sim time	4	It is the amount of time to forward-simulate trajectories in seconds.
Controller Frequency	10	It is the frequency at which the controller is called in Hz.
Pdist Scale	0.5	It is the weighting for how much the controller should stay close to the path it was given
Yaw Goal Tolerance	0.1	It help define how close the robot's pose can be to the goal position
XY Goal Tolerance	0.02	Same description as Yaw Goal Tolerance

4 Discussion

UdacityBot was able to reach the destination goal with no problem. However the route taken was not that efficient, it could have been shorten with more tuning of the parameters.

Since AMCL doesn't rely on landmarks but on laser based maps, the laser scans and transform message to output pose estimates, UdacityBot would do fine if the kidnapping robot problem occurred. Where the robots would be positioned in an random location and the ACML would be able to adapt to the number of particles used to gain certainty of where the robots are located.

FouliexBot is based on UdacityBot and it may come to one surprise that changing the robot shape from a rectangle shape to a square shape change the robot drastically that one would need to change the parameters.Furthermore after changing the parameters FouliexBot was not at the same performance level as UdacityBot.

5 Future Work

List of future work that can be done to improve both robots:

1) Add a LIDAR unit. It will create a more complete map of the world around the robot
2) Add a Laser GPU gazebo component to reduce the load from the CPU.
3) Test the Kidnapping problem on UdacityBot and tune them if needed.
4) Deploy the UdacityBot to a Jetson TX2 and test that the hardware configuration has an adequate processing power such as memory and CPU power.

6 Result images

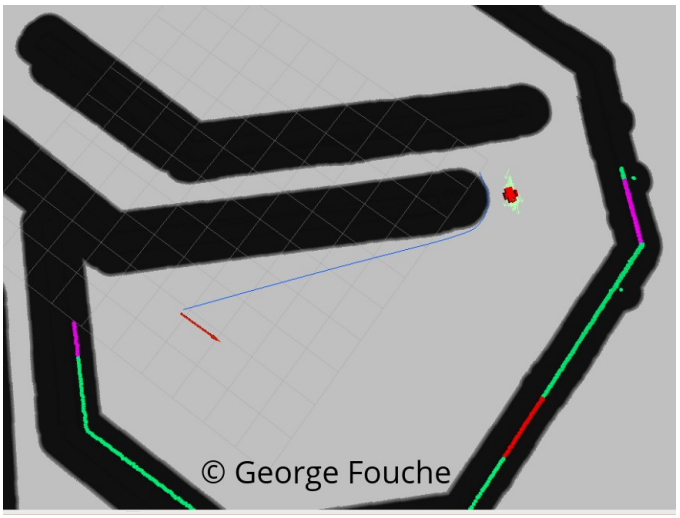

Fig. 5. Showing how one of the robot was able to make the turn successfully

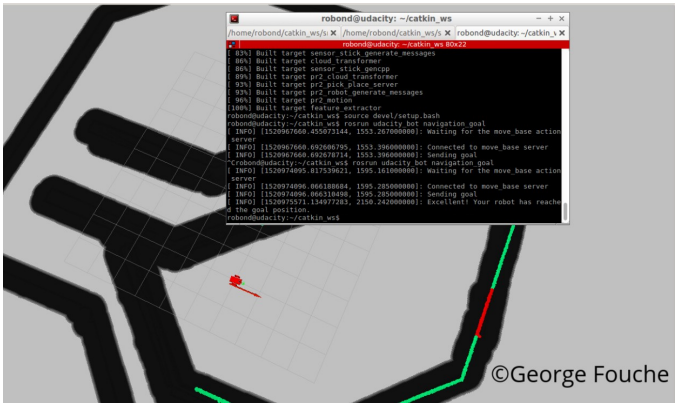

Fig. 6. Showing the goal is met for one of the robot

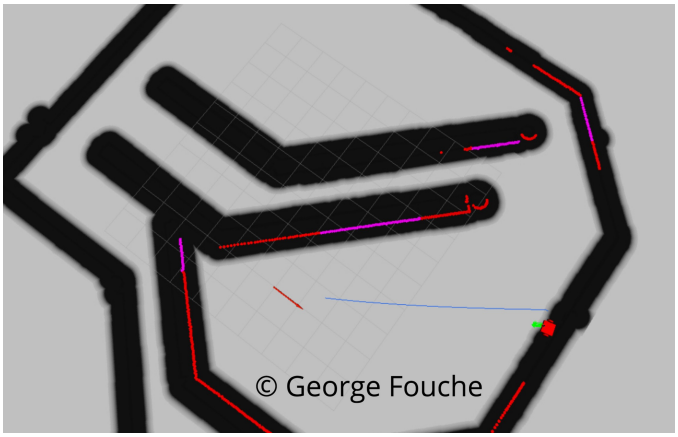

Fig. 7. FouliexBot crashing after making the turn Goal the turn