Robotics Software Engineer Nanodegree: Localization Project

Manuel Huertas López

Abstract—This project involves to accurately localize a mobile robot insided a provided map in Gazebo and RViz simulation enviroments. The goal includes to build a robot model in Gazebo and utilize packages like AMCL and the Navigation Stack to succefully identifed the position of the robot in the world. As a final test the robot will be commaned to a specific location and the navigation stack will determinate the route to follow. As part of the work it is neccessary to explore, add, and tunning specific parameters corresponding to each package to achieve the best possible localization result.

Index Terms—Robot, Udacity, Monte Carlo Filter, Kalman Filter, Localization,ROS.

1 Introduction

Mobile robot localization is the process to determine where a robot is located with respect to its environment. It is one of the fundamental parts required by a robot to achieve autonomous navigation.

In its simplest form, position tracking, the initial robot pose is known and it is only necessary to correct incremental errors in the robot odometry. More challenging is the global localization problem, where the robot must determinate its initial pose by itself. Finally, in the kidnapped robot problem, the robot is tele-ported to other position without being told. The kidnapped problem can be used to teach the robot how to recover from a system fail, where the robot previous knowledge is complete lost.

Localization can be used in so many applications ranging from robots to clean the floors to robot to explore other planets.

The localization algorithms must overcome some difficulties like the errors in sensor measurement; like sensors to measure the distances in scene, and errors in the control. The algorithm used for the project use Bayesian probability to evaluate the probability of a hypothesis, where the robot is located, base on the observation of the environment.

2 Background

In the present work Kalman filter and Adaptative Monte Carlo have been evaluated.

They are both base on probability distribution of the states over measurements using a Bayes filter.

Adaptive Monte Carlo algorithm has been selected in order to solve the Localization problem. AMCL, a particle filter, has got so many advantages over other solutions: the error and noise does not need to follow a Gaussian distribution, the computational needs can be easily adapted, it is a very robust and fast algorithm. [**?**]

2.1 Kalman Filters

Kalman filter was invented by Rudolf Kalman and it was used to solve non-linear problem of the trajectory estimation of the Apollo program. It helped to the Apollo to orbit around the moon.

Kalman filter is an estimation algorithm used to estimate the value of a variable as the data is being collected. It can take data, from sensors, with a lot of uncertainty or noise in measurements and provides a very accurate estimate of the real value. It can do fast and without needing too much computer power.

Kalman filter can be used for local localization problem where both the odometry and movement contains a lot of noise and uncertain. The Kalman filter is continuously iterating two steps: measurement update where measurement is used to update the robot's position, and state prediction where the information about current state is used to predict the future pose. At the beginning an initial guess is used. After a few iterations the measurement converge to the real value.

The Kalma filter assumes a Gaussian distribution of the noise an uncertain and it is only valid for linear system.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{1}$$

Nevertheless, a variation of the algorithm, Extended Kalman filter can be used for non-linear system. The EKF linearise the original non-linear filter dynamics around the previous estimates.

2.2 Particle Filters

The Monte Carlo localization is the most popular Localization algorithm in robotics. It can solve the local an global localization problems. The robot will navigate inside a known map collecting information using range sensors. MCL uses these sensor measurement to keep track of the robot pose.

In MCL a set of particles are initially spread randomly and uniformly throughout the entire map. These virtual particles represents the hypothesis of where the robot might be. In a 2D map, each particle has got a position , orientation and weight value. The importance of a particle depends on the weight, the bigger the more accurate and during the re-sampling process they are more likely to survive. After

several iterations of the Monte Carlo algorithm the particles will converge and estimate the robot's pose.

The MCL uses a Bayes filtering to estimate a probability density over the state space conditioned by the measurement, this probability density is called belief:

$$Bel(Xt) = P(Xt|Z1...t) \quad (2)$$

Xt are the state of robot and Z are the measurement from time 1 up to time t. The belief is estimated recursively, the initial guess uses a randomly and uniformly distribution over the entire map.

The MCL has got two sections. In every iteration the algorithm takes: previous belief, the actuation commands and the sensor measurement as input. The hypothetical state is computed whenever the robot moves, the particles weight are computed using the latest sensor measurement. Secondly the particles with high probability, the ones that fetch better the measurement reading, will survive and the others will die. The algorithm will output the new belief of the next iteration. After few iterations the survivors particles will represent the real robot's pose.

2.3 Comparison / Contrast

MCL presents many advantages over EFK:

- MCL is easier to program than EFK
- In the EFK algorithm the noise error and uncertain must follow a Gaussian distribution, MCL is not tied to the measure noise distribution.
- MCL can control the computational memory and resolution of the solution by changing the number of particles distributed.

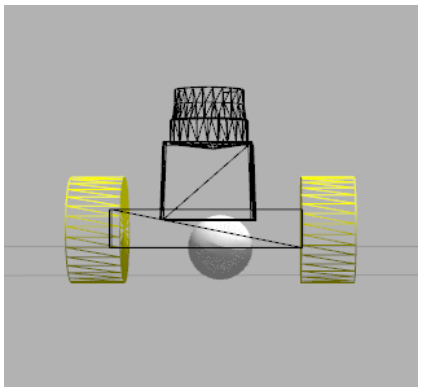
In the present work MCL particle filter is going to be used.

3 Simulations

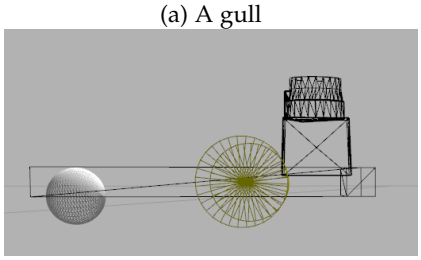
This section should discuss the performance of robots in simulation. Items to include are the robot model design, packages used, and the parameters chosen for the robot to properly localize itself. The information provided here is critical if anyone would like to replicate your results. After all, the intent of reports such as these are to convey information and build upon ideas so you want to ensure others can validate your process. You should have at least two images here: one that shows your standard robot used in the first part of the project, and a second robot that you modified / built that is different from the first robot. Remember to watermark all of your images as well.

3.1 Achievements

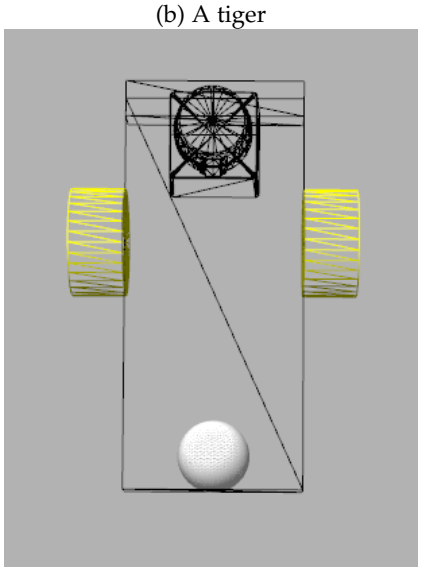
You should describe what you achieved for localization in the project with the benchmark model and your own model. Includes charts and graphs show how parameters affect your performance.



(a) A gull



(b) A tiger



(c) A mouse

Fig. 1: Robot desing view

3.2 Benchmark Model

3.2.1 Model design

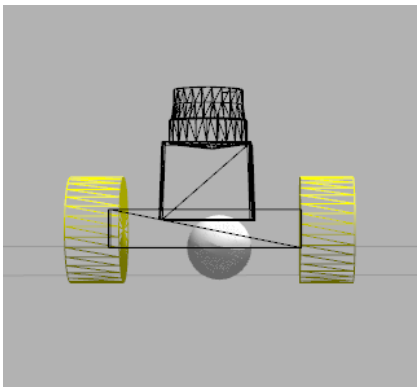
The Robot's design considerations should include: the size of the robot, the layout of sensors. This information can be shown in the form of a chart / table.

3.2.2 Packages Used

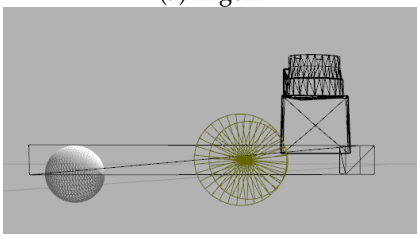
The packages used in the project should be specified as well as the topics received and published; the services it used and provided should also be addressed.

3.2.3 Parameters

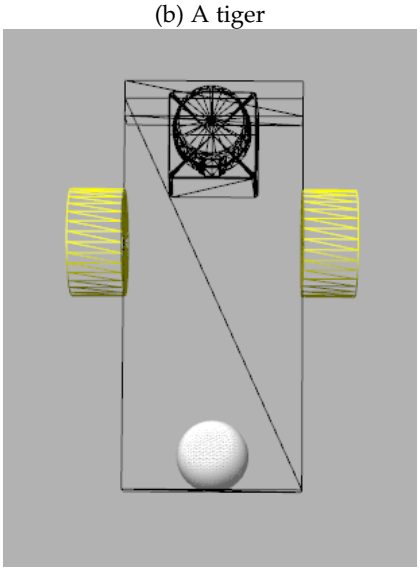
Localization parameters in the AMCL node should be described, as well as move_base parameters in the configuration file. You should be able to clearly demonstrate your understanding of the impact of these parameters.



(a) A gull



(b) A tiger



(c) A mouse

Fig. 2: Robot desing view

3.3 Personal Model

3.3.1 Model design

- Chassis:
- Wheels:
- Back caster:
- Lidar:
- Camera:

3.3.2 Packages Used

- gazebo_ros:
- rviz:
- map_server:
- tf:
- amcl:
- move_base:

3.3.3 Localization parameters: Overall Filter

The following configuration parameters were choosen due to power limitations of the computer where the package was deployed:

- min_particles[100]: Minimum allowed number of particles spread around the scene.
- max_particles[5000]: Maximum allowed number of particles spread around the scene.
- transform_tolerance[0.5]: Time with which to post-date the transform that is published, to indicate that this transform is valid into the future.
- gui_publish_rate[1]: Maximum rate (Hz) at which scans and paths are published for visualization.

The following configuration parameters are related about how to recover from stuck:

- recovery_alpha_slow[0.001]: Exponential decay rate for the slow average weight filter, used in deciding when to recover by adding random poses.
- recovery_alpha_fast[0.1]: Exponential decay rate for the fast average weight filter, used in deciding when to recover by adding random poses.

The following parameters give and estimation of the initial pose, mean and covariance, of the robot in the world; previous to any movement:

- initial_pose_x[0.0]: Initial pose mean (x), used to initialize filter with Gaussian distribution.
- initial_pose_y[0.0]: Initial pose mean (y), used to initialize filter with Gaussian distribution.
- initial_pose_a[0.0]: Initial pose mean (yaw), used to initialize filter with Gaussian distribution.
- initial_cov_xx[0.25]: Initial pose covariance (x*x), used to initialize filter with Gaussian distribution.
- initial_cov_yy[0.25]: Initial pose covariance (y*y), used to initialize filter with Gaussian distribution.
- initial_cov_aa[0.07]: Initial pose covariance (yaw*yaw), used to initialize filter with Gaussian distribution.

3.3.4 Localization parameters: Laser model

- laser_likelihood_max_dist[2.0]: Maximum distance to do obstacle inflation on map, for use in likelihood_field model.
- laser_model_type[likelihood_field]: Which model to use, either beam, likelihood_field.

3.3.5 Localization parameters: Odom parameters

- odom_model_type[diffcorrected]: Which model to use, either ”diff”, ”omni”, ”diff-corrected” or ”omni-corrected”. In our case the robot is diff type, the corrected include a bug correction.

The following parameters are about the noise in the odemotry. Because we are using a Bayesian Filter the probability of the true value of the odometry can be tunned using these parameters. The values where obtanined by trail and error.

- odom_alpha1[0.005]: Specifies the expected noise in odometry’s rotation estimate from the rotational component of the robot’s motion.

- odom_alpha2[0.005]: Specifies the expected noise in odometry's rotation estimate from translational component of the robot's motion.
- odom_alpha3[0.010]: Specifies the expected noise in odometry's translation estimate from the translational component of the robot's motion.
- odom_alpha4[0.010]: Specifies the expected noise in odometry's translation estimate from the rotational component of the robot's motion.

3.3.6 Move base parameters: costmap_common_params

The navigation stack uses costmaps to store information about the obstacles in the world.

- obstacle_range[2.5]: The robot will only update its map with information about obstacles that are within 2.5 meters of the base.
- raytrace_range[3.0]: This parameter determines the range to which we will raytrace freespace given a sensor reading. The robot will attempt to clear out space in front of it up to 3.0 meters away given a sensor reading.
- robot_radius[0.11]: The robot radious is 11cm. It is used to avoid passing the robot through narrow places where it can get stucked.
- inflation_radius[0.2]: The robot will treat all paths that stay 0.2 meters or more away from obstacles as having equal obstacle cost.

3.3.7 Move base parameters: local_costmap_params

The local costmap is used by the local planner to generate a short-term plan. The local costmap uses the odometry information.

- robot_base_frame:
- update_frequency[2]: The frequency, in Hz, at which the costmap will run its update loop.
- publish_frequency[2]: The rate, in Hz, at which the costmap will publish visualization information.
- width[20.0]: The width of the costmap.
- height[20.0]: The height of the costmap.
- resolution[0.05]: The resolution of the costmap.
- static_map[false]: The costmap should not initialize itself based on a map served by the map_server, for the local costmap the odometry is used.
- rolling_window:
- transform_tolerance:

3.3.8 Move base parameters: global_costmap_params

- robot_base_frame:
- update_frequency:
- publish_frequency:
- width:
- height:
- resolution:
- static_map:
- rolling_window:
- transform_tolerance:

3.3.9 Move base parameters: base_local_planner_params

- holonomic_robot:
- sim_time:
- meter_scoring::
- max_vel_x:
- min_vel_x:
- max_vel_theta:
- min_vel_theta:
- acc_lim_x:
- acc_lim_y:
- acc_lim_theta:

4 RESULTS

Present an unbiased view of your robot's performance and justify your stance with facts. Do the localization results look reasonable? What is the duration for the particle filters to converge? How long does it take for the robot to reach the goal? Does it follow a smooth path to the goal? Does it have unexpected behavior in the process?
For demonstrating your results, it is incredibly useful to have some watermarked charts, tables, and/or graphs for the reader to review. This makes ingesting the information quicker and easier.

4.1 Localization Results

4.1.1 Benchmark

4.1.2 Student

4.2 Technical Comparison

Discuss the difference of the layout, parameters, performance etc. between the benchmark robot and your robot. It is acceptable for your custom robot to perform worse than the provided robot. The focus is on learning and understanding, not performance.

5 DISCUSSION

This is the only section of the report where you may include your opinion. However, make sure your opinion is based on facts. If your robot performed poorly, make mention of what may be the underlying issues. If the robot runs well, which aspects contribute to that? Again, avoid writing in the first person (i.e. Do not use words like ″I″ or ″me″). If you really find yourself struggling to avoid the word ″I″ or ″me″; sometimes, this can be avoid with the use of the word one. As an example: instead of : ″I think the robot cannot localize itself because the sensor does not provide enough information for localization″ try: ″one may believe the localization performance is poor because the sensor layout is not able to provide enough information for localization″. They say the same thing, but the second avoids the first person.

5.1 Topics

- Which robot performed better?
- Why it performed better? (opinion)
- How would you approach the 'Kidnapped Robot' problem?
- What types of scenario could localization be performed?
- Where would you use MCL/AMCL in an industry domain?

6 CONCLUSION / FUTURE WORK

This section is intended to summarize your report. Your summary should include a recap of the results, did this project achieve what you attempted, how would you deploy it on hardware and how could this project be applied to commercial products? For Future Work, address areas of work that you may not have addressed in your report as possible next steps. This could be due to time constraints, lack of currently developed methods / technology, and areas of application outside of your current implementation. Again, avoid the use of the first-person.

6.1 Modifications for Improvement

Examples:

- Base Dimension
- Sensor Location
- Sensor Layout
- Sensor Amount

6.2 Hardware Deployment

1) What would need to be done?
2) Computation time/resource considerations?