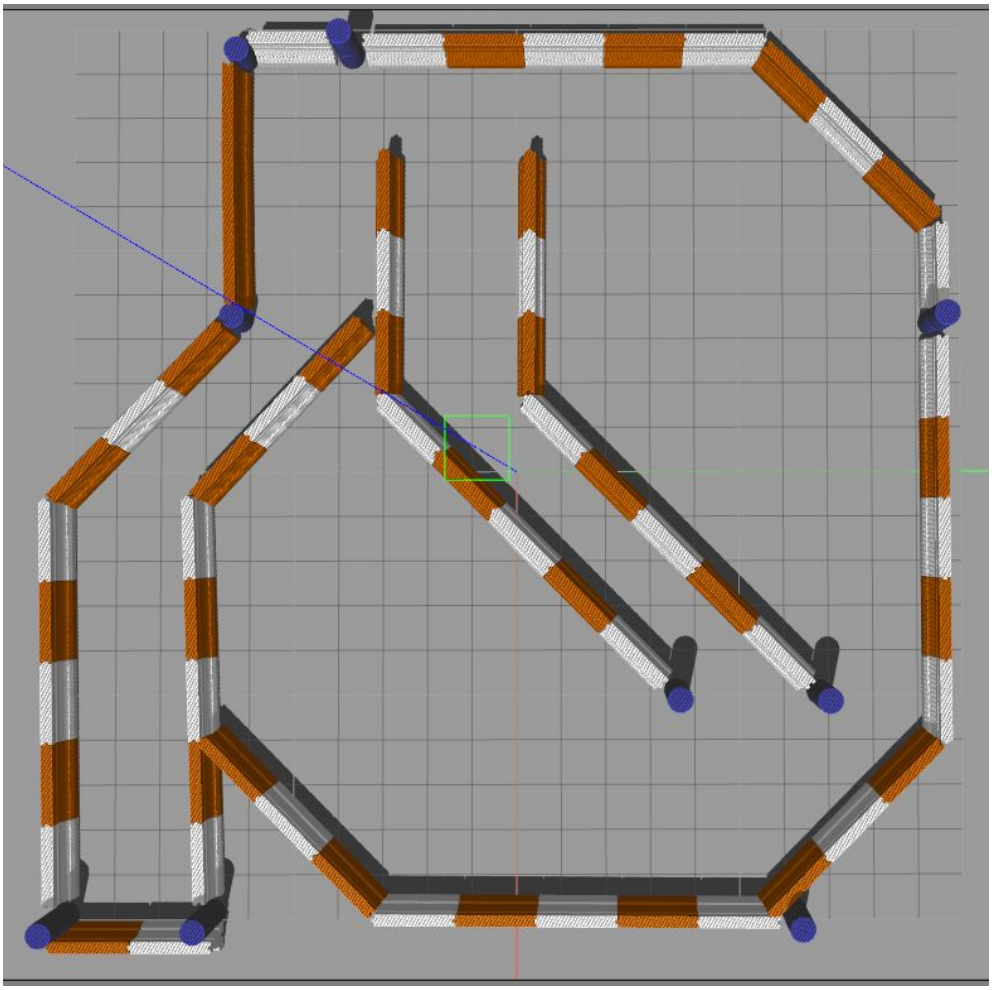
Project: Where Am I?

Abstract

Robot localization is the process of determining where a mobile robot is located with respect to its environment, we are given a robot and asked to build another one, after that to configure them with the AMCL package to localize themself.

Introduction

Robot localization provides an answer to the question: Where is the robot now? A reliable solution to this question is required for performing useful tasks, as the knowledge of current location is essential for deciding what to do next, the project contains two parts:
1- we are given a robot model and asked to use the AMCL package to localize the robot and use navigation stack package to move the robot to a given goal.
2- we have to build our own model and asked to repeat the experiment on our robot.



The world in which our robot should localize itself and move to the goal

Background

Knowledge of the reliability of the location estimate plays an important role in the decision making processes used in mobile robots as fatal consequences may follow if decisions are made assuming that the location estimates are perfect when they are uncertain. Bayesian filtering is a powerful technique that could be applied to obtain an estimate of the robot location and the associated uncertainty. Both extended Kalman filter (EKF) and particle filter provide tractable approximations to Bayesian filtering

Kalman Filters

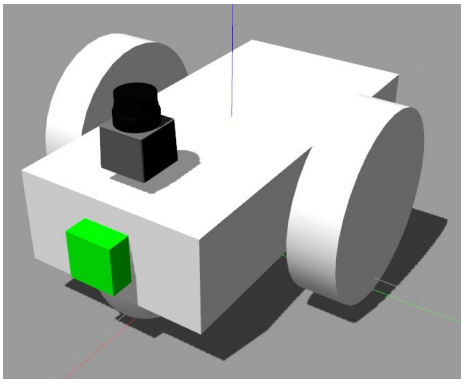
Kalman filter is a technique for filtering and prediction in linear Gaussian systems.

Particle Filters

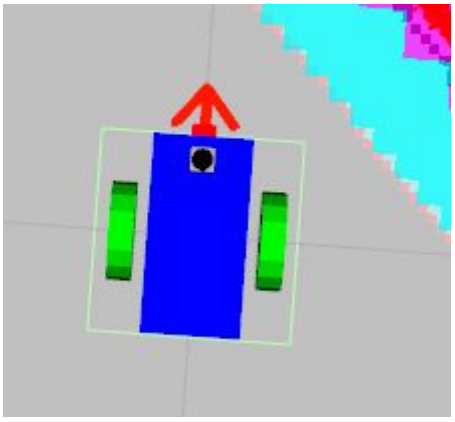
In the particle filter localization (also known as Monte Carlo localization) a weighted set of robot location estimates, termed as particles, is used to describe the probability distribution of the robot location. In the particle filter, each particle in effect provides a guess as to the location of the robot.

Kalman Filter	Particle Filter
Gaussian probability distribution	Any probability distribution
Unimodal	Multimodel
Local localization	Local and global localization
Continuous state space	Discrete state space

Models

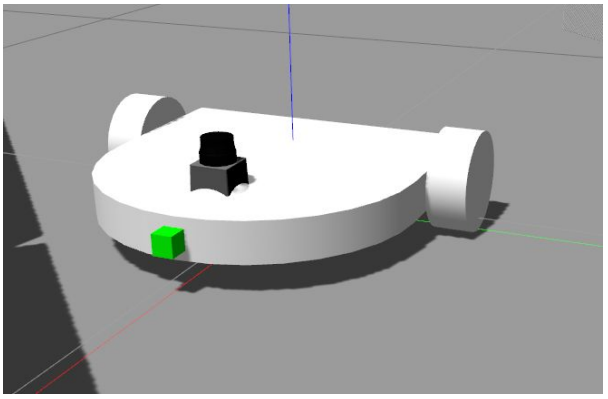


The given model

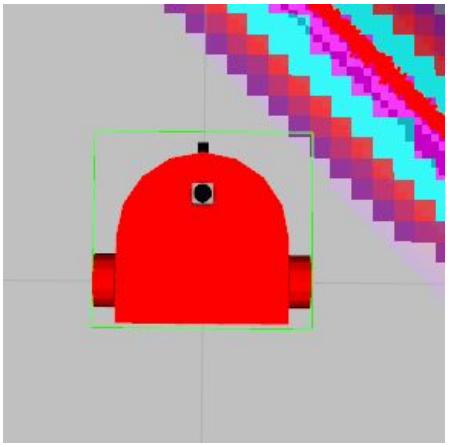


The footprint of the given robot

footprint: [[0.2, 0.2], [0.2, -0.2], [-0.2, -0.2], [-0.2,0.2]]



The customized model



The footprint of the customized robot

footprint: [[0.35, 0.25], [0.35, -0.25], [-0.1, -0.25], [-0.1,0.25]]

Parameters

The parameters for both robots are identical except for the footprint

base_local_planner_params	
holonomic_robot: false	Differential-drive robot
meter_scoring	The values are in meter
yaw_goal_tolerance	To prevent the robot from unnecessary rotation around the goal
xy_goal_tolerance	To prevent the robot from unnecessary rotation around the goal

costmap_common_params	
obstacle_range	This parameter sets the maximum distance, relative to the robot, an object/obstacle can be added to the map.
raytrace_range	This parameter sets the maximum range in meters at which to raytrace objects/obstacles from the map.
transform_tolerance	This parameter specifies the de- lay in transform data that is still acceptable, set it to 0.1 because my PC isn't fast enough to set to smaller value
inflation_radius	This parameter describes the inflation radius in meters from the robot center, It is used to avoid collisions, its value is 0.2 because it's approximately the radius of the robots

The values in global_costmap_params and local_costmap_params was set experimentally, the width and height of local_costmap was set to 3.0 because higher values was preventing the robot form following the local path.

Results

The following two links show how the robots was able to go to goal successfully.

The customized model	The given model

Discussion

The results show an Adaptive Monte Carlo localization
implementation of two robots in a simulation environment. Both robots
were able to localize autonomously in the given map
and successfully reach the set target position while simultaneously avoiding all obstacles.

AMCL would not work too well for the kidnapped robot problem, but we can use another algorithms using visual features to recognize places (visual features are usually more distinguishing than laser features).

Conclusion / Future work

1- Testing it on a real robot.
2- Writing my own path planner, trajectory planner.
3- Play with more parameter to see if it enhances the implementation.