Laboratory 2
Mobile Robot Simulation

CSC 232 / ECE 232 / CSC 432 / ECE 437 - Autonomous Mobile Robots

February 8th, 2016 - February 24th, 2016

Academic Honesty

The following laboratory exercises and and the laboratory report must be performed in accordance with the University of Rochester's Academic Honesty Policy. You may discuss the laboratory exercises with others, however all of the code, data, and material contained within the report must be your own in accordance with the laboratory instructions. Please type or write "I affirm that I have not given or received any unauthorized help on this assignment, and that all work is my own." at the beginning of the laboratory report. Please contact Professor Howard at thomas.howard@rochester.edu if you have any questions about the academic honesty policy for this laboratory.

Report Due Date

This document describes a two week long laboratory due on February 24th, 2016. The laboratory report shall be written in LaTeX, Word, Pages, or the text editor of your choice and submitted as a PDF with a tarball containing the source code through the course website on Blackboard. The laboratory report shall contain a description of the activities taken in each step of the laboratory and answer all questions posed at the end of this document.

Overview

Laboratory 2 will build upon the code that you developed in Laboratory 1 to apply sampling-based methods for probablistic motion models to simulate the motion of a TurtleBot robot using ROS. By the end of this two-part laboratory, you should be able to:

- Modify a CMakeLists.txt file to create another executable program
- Write a process that contains multiple a publishers and subscribers
- Convert spatial rotations between Quaternions and Euler Angles
- Sample from a zero-mean normal distribution
- Simulate the uncertain motion of a mobile robot using the velocity motion model

Requirements

Laboratory 2 will require that you implement two programs: a *navigator* and a *simulator*. The *navigator* serves as the method that controls the commands to the robot, and should be based on the code that was written in Laboratory 1. The *simulator* operates as a drop-in replacement for the TurtleBot hardware that listens and publishes to the same channels as the robot. Specifically, the *simulator* must have a callback function that listens for ROS geometry_msgs::Twist messages on the "/cmd_vel_mux/input/navi" channel and ROS std_msgs::Empty messages on the "/mobile_base/commands/reset_odometry" channel and periodically publishes ROS nav_msgs::Odometry messages on the "/odom" channel. When the geometry_msgs::Twist messages message and std_msgs::Empty message are received, the simulator must update the commanded linear and angular velocities and reset the simulator position and orientation respectively. An illustration of the message passing between the two required programs is illustrated in Figure 1.

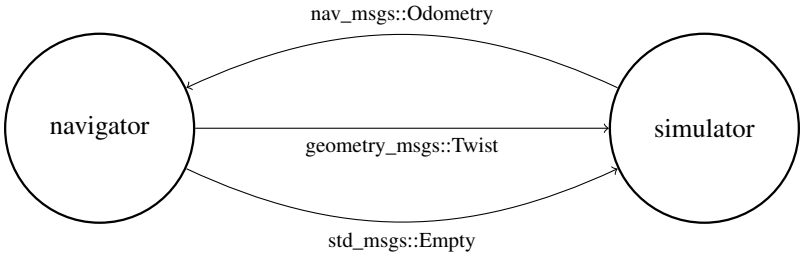

Figure 1: A diagram of the message passing between the *navigator* and *simulator* programs in Laboratory 2

The *navigator* program is required to execute any of the five velocity profiles from Laboratory 1 that can be controlled through a command-line argument defined through the gengetopt (.ggo) file that is associated with the main program (.cc) file. The *navigator* is required to record the pose and velocity of the robot during each of these control sequences and include plots of the position (x and y), yaw (θ), linear velocity (v_x), and angular velocity (ω_z) as a function of time. The *navigator* is based on the main program that was written for Laboratory 1, and should only require minor (if any) changes. Specifically, the *navigator* must have a callback function that listens for ROS nav_msgs::Odometry messages on the "/odom" channel and have a callback that records the simulated robot pose and commands as a function of time. For reference, the five velocity profiles from Laboratory 1 are listed below:

1. $v_x\left(t\right) = 0.25 \frac{m}{sec}$ and $\omega_z\left(t\right) = 0 \frac{rad}{sec}$ for 4 seconds

2. $v_x\left(t\right) = 0.25 \frac{m}{sec}$ and $\omega_z\left(t\right) = 1.0 \frac{rad}{sec}$ for 4 seconds

3. $v_x\left(t\right) = 0.25 \frac{m}{sec}$ and $\omega_z\left(t\right) = -1.0 \frac{rad}{sec}$ for 4 seconds

4. $v_x\left(t\right) = 0.25 \sin\left(t\right) \frac{m}{sec}$ and $\omega_z\left(t\right) = 0 \frac{rad}{sec}$ for 10 seconds

5. $v_x\left(t\right) = 0 \frac{m}{sec}$ and $\omega_z\left(t\right) = 1.0 \sin\left(t\right) \frac{rad}{sec}$ for 10 seconds

The *simulator* program is required to simulate the behavior of the TurtleBot robot by replicating the functionality of the hardware. The motion simulation of the *simulator* is required to use the algorithm for the velocity-based motion model that can be found in Table 5.3 in Probabilistic Robotics. The implementation of this algorithm is to use the normal distribution with coefficients prescribed in the question section.

Code Checklist

☐ ***simulator*** **nav_msgs::Odometry publisher for the "/odom" channel**
☐ ***simulator*** **std_msgs::Empty subscriber for the "/mobile_base/commands/reset_odometry" channel**
☐ ***simulator*** **geometry_msgs::Twist subscriber for the "/cmd_vel_mux/input/navi" channel**
☐ ***simulator*** **function to convert yaw to Quaternion**
☐ ***simulator*** **function to sample from a normal distribution**
☐ ***simulator*** **function to evaluate the probabilistic velocity motion model**

☐ ***navigator*** **nav_msgs::Odometry subscriber for the "/odom" channel**
☐ ***navigator*** **std_msgs::Empty publisher for the "/mobile_base/commands/reset_odometry" channel**
☐ ***navigator*** **geometry_msgs::Twist publisher for the "/cmd_vel_mux/input/navi" channel**
☐ ***navigator*** **function to convert Quaternion to yaw**

Questions

1. Using Matlab, plot the simulated robot position (x and y), yaw (θ), linear velocity (v_x), and angular velocity (ω_z) recorded from your program as a function of time for each of the five open-loop motion control sequences in the Requirements section using the following coefficients for the zero-mean normal distributions:

   (a) $\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 \end{bmatrix} = \begin{bmatrix} 0.01 & 0.01 & 0.1 & 0.1 & 0.01 & 0.01 \end{bmatrix}$

   (b) $\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.01 & 0.01 \end{bmatrix}$

2. Using Matlab, plot a fixed goal position ($goal_{x,y} = (5.0\,m, 0.0\,m)$) in the local frame of the robot as a function of time for each of the ten datasets from the previous question.

Reference equations on back of sheet.

Quick Reference

This section provides a quick reference for some of the mathematics that are necessary to complete the laboratory. These equations may also be found in Probabilistic Robotics or the books held on reserve in the library. For spatial rotations, note the relationships between rotation matrices, the rotation matrix for a heading (θ) rotation, and the Unit Quaternion for Spatial Rotations.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{1}$$

$$R(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ \sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$Q = \{\eta, \epsilon\} \tag{3}$$

$$\eta = \frac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1} \tag{4}$$

$$\epsilon = \frac{1}{2}\begin{bmatrix} \text{sgn}(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \\ \text{sgn}(r_{13} - r_{31})\sqrt{r_{22} - r_{33} - r_{11} + 1} \\ \text{sgn}(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \end{bmatrix} \tag{5}$$

$$R = \begin{bmatrix} 2\left(\eta^2 + \epsilon_x^2\right) - 1 & 2(\epsilon_x\epsilon_y - \eta\epsilon_z) & 2(\epsilon_x\epsilon_z + \eta\epsilon_y) \\ 2(\epsilon_x\epsilon_y + \eta\epsilon_z) & 2\left(\eta^2 + \epsilon_y^2\right) - 1 & 2(\epsilon_y\epsilon_z - \eta\epsilon_x) \\ 2(\epsilon_x\epsilon_z + \eta\epsilon_y) & 2(\epsilon_y\epsilon_z + \eta\epsilon_x) & 2\left(\eta^2 + \epsilon_z^2\right) - 1 \end{bmatrix} \tag{6}$$

For the velocity motion model you will need to sample from a normal distribution, which can be approximated with the following function:

$$\frac{1}{2}\sum_{i=1}^{12} \text{rand}(-b, b) \tag{7}$$

The velocity motion model itself can be computed as follows:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}}{\hat{\omega}}\sin(\theta) + \frac{\hat{v}}{\hat{\omega}}\sin(\theta + \hat{\omega}\Delta t) \\ \frac{\hat{v}}{\hat{\omega}}\cos(\theta) - \frac{\hat{v}}{\hat{\omega}}\cos(\theta + \hat{\omega}\Delta t) \\ \hat{\omega}\Delta t + \hat{\gamma}\Delta t \end{pmatrix} \tag{8}$$