Assignment 2: Localization and mapping

ETH Zurich

Due: Thursday, 14.11.2013 at 13:00

”Boom! Boom!” An earthquake. The building is shaking vigorously. You wander around the building in search of an exit. Then, lights go off. Pitch black darkness. You feel your best bet for survival is stay where you are and wait for your robot to come for a rescue. After a while, silence. Your robot finally enters the building and starts looking for you. Can it get to you and help you get out of this building safely?

1 ROS

In this assignment, you will create a navigation node in ROS and implement particle filter based localization and mapping algorithms. If you have not already done so, do the ROS tutorials (http://wiki.ros.org/ROS/Tutorials) and learn how to create a ros package and write a publisher and subscriber.

In addition, the following will be useful in completing the assignment.

1.1 tf

tf is a package for keeping track of different coordinate frames over time. It maintains the relationship between coordinate frames in a tree structure and lets you transform points, vectors, etc. between frames. In this assignment, tf will be particularly useful in transforming sensory information from the PrimeSense to the robot base frame and the global frame. Learn more about tf package here:http://wiki.ros.org/tf.

1.2 RViz

RViz is a 3D visualization tool for ROS. Rviz lets you subscribe to various topics such as odometry, map, laser sensor, etc., and visualize them on the screen. For every subscribed topic, there must be a transformation tree between the fixed frame and the target frame (the frame of the topic), and this transformation is defined using tf. Learn more about RVis here: http://wiki.ros.org/rviz. In particular, follow the user guide (http://wiki.ros.org/rviz/UserGuide) and learn to use the visualization tool.

1.3 map_server

map_server is a ROS node that offers map data as a ROS service. It also provides the map_saver command-line utility, which allows dynamically generated maps to be saved to file. Learn more about map_server here: http://wiki.ros.org/map_server

1.4 depthimage_to_laserscan

PrimeSense 1.09 RGB-D camera that provides (x, y, g) and (r, g, b) values for each pixel it sees. In this assignment, you will use the sensor to get ”laser scans” from the sensor. depthimage_to_laserscan package simulates laser scans from the sensor by returning depth value of a single scan line.

To install, open a terminal and run the following command:

```
sudo apt−get install ros−groovy−depthimage−to−laserscan
```

More information on the package is found here: http://wiki.ros.org/depthimage_to_laserscan

1.5 common_msgs

common_msgs contains messages that are widely used by other ROS packages, including geometry_msgs, nav_msgs, and sensor_msgs. geometry_msgs provides messages for common geometric primitives such as points, vectors, and poses. nav_msgs defines navigation-related messages such as odometry and map. sensor_msgs defines messages for commonly used sensors including range and laser sensors. Learn more about common_msgs here: http://wiki.ros.org/common_msgs. In particular, learn about sensor_msgs/LaserScan and nav_msgs/OccupancyGrid.

2 Localization

2.1 Background

Mobile robot localization, also known as position estimation, is the process of determining a robot's pose with respect to a given map of an environment. The localization problem is one of the fundamental perception problems in robotics. The ability to localize is critical to all robots that need to interact with their environment.

Particle filter localization [2] is a localization method based on particle filter. Particle filter is a nonparametric filter in which a distribution is represented by a set of random samples drawn from the distribution. The samples are called *particles*, $X_t = \{x_t^1, x_t^2, ..., x_t^M\}$, and in localization, they represent concrete instantiations of the robot state $x_t^m = (x, y, \theta)$ at time t. The number of particles, M, influence the accuracy of the localization algorithm.

Algorithm 1: Particle filter localization algorithm [1]

```
Data: $X_{t-1}$: a set of particles (robot states) at time $t-1$
      $u_t$: the robot control at time $t$
      $z_t$: the sensor measurement at time $t$
Result: $X_t$: a set of particles at time $t$
begin
    $\bar{X}_t = X_t = \emptyset$
    for $m = 1$ to $M$ do
        $x_t^{[m]} =$ motion_update($u_t, x_{t-1}^{[m]}$)
        $w_t^{[m]} =$ sensor_update($z_t, x_t^{[m]}$)
        $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
    for $m = 1$ to $M$ do
        $x_t^{[i]} =$ resample($\bar{X}_t$)
        $X_t = X_t + x_t^{[i]}$
end
```

The basic algorithm is shown in Algorithm 1. The algorithm takes a set of particles from the previous time $t-1$ and the control input and current measurement as input and produces a new set of particles. The **motion_update** function updates a previous particle $x_{t-1}^{[m]}$ by moving it according to the control u_t and adding some Gaussian noise. The **sensor_update** function computes the weight for each particle by computing the probability of the current measurement reading given the robot pose. Lastly, the **resample** function takes the updated particles and their weights and draws new particles such that the probability of a particle $x_t^{[i]}$ being drawn is proportional to its weight $w_t^{[i]}$.

Particle filter localization is able to represent wide range of distributions and thus solve global localization problem. In addition, if some particles are replaced by random samples, it is able to handle kidnapped robot problem in which a robot can find its way out of a wrong pose prediction.

Some useful tutorials on particle filter include:

- `http://robots.stanford.edu/papers/fox.mcmc-book.ps.gz`
- `http://www.cim.mcgill.ca/~yiannis/particletutorial.pdf`

2.2 Task

Implement a particle filter localization algorithm using PrimeSense as the sensor. Play with the number of particles and sample distributions.

3 Grading

3.1 In-class demonstration (20 points)

On Thursday, 14.11.2013, during the exercise session, your implementation will be tested on its ability to localize.

You will be given a map of an environment as a png file and a destination. You need to demonstrate that your robot can localize itself without knowing its starting pose and go to the desired destination.

The grading scheme is as follows:

- Accuracy: Position control to the goal
  - Error of more than 60%: 0 point
  - For every 3% reduction of error: +1 point
  - Error of 3% or less: 20 points
- Restart: You can restart the robot once.

3.2 Software quality (20 points)

On the due date at the due time (Thursday, 14.11.2013, 13:00), we will collect your code via your svn repository. Every file that should be considered for grading must be in the repository at that time.

- Choice of abstraction and relations (6 points)
- Correctness of implementation (8 points)
- Extendibility and reusability (4 points)
- Comments and documentation, including "README" (2 points)

References

[1] Thrun, S., Burgard, W., Fox, D. 2005. Probabilistic Robotics MIT Press. Chapter 8.

[2] Dellaert, F., Fox, D., Burgard, W., Thrun, S., 1999. Monte Carlo Localization for Mobile Robots, ICRA99.