Teaching Localization in Probabilistic Robotics

Fred G. Martin, James Dalphond, and Nat Tuck
University of Massachusetts Lowell
Computer Science
1 University Avenue
Lowell, Massachusetts 01854

Abstract

In the field of probabilistic robotics, a central problem is to determine a robot's state given knowledge of a time series of control commands and sensor readings. The effects of control commands and the behavior of sensor devices are both modeled probabilistically. A variety of methods are available for deriving the robot's belief state, which is a probabilistic representation of the robot's true state (which cannot be directly known). This paper presents a series of five weekly assignments to teach this material at the advanced undergraduate/graduate level. The theoretical aspect of the work is reinforced by practical implementation exercises using ROS (Robot Operating System), and the Bilibot, an educational robot platform.

Background

Over the last twenty years, probabilistic robotics has become a dominant and successful theoretical framework for creating highly competent autonomous robotic systems (Thrun 2002; 2010). Based on (Thrun, Burgard, and Fox 2005), the reference text in the field, we have developed a series of assignments to introduce students robot localization using probabilistic methods.

The paper briefly introduces the relevant theory, and then presents our student assignments. Using ROS (Robot Operating System, `www.ros.org`), students implement code that controls a simulated robot in a virtual world. We introduce noise in the simulated robot's controls and sensors that mirrors noise found in the real world. Our students subsequently load their control programs onto the Bilibot (`bilibot.com`), an educational robotics platform based on the iRobot Create.

Introduction

The essential theory for probabilistic robotics is based on hidden Markov models (HMMs). The robot's true state $\mathbf{X}$ is the hidden variable. For the localization problem of a ground robot, the state may consist of just three components: its (x, y) location in the world and its rotational orientation θ.

Influenced by the hidden state, sensor readings produce probabilistic emissions $\mathbf{Z}$. The robot's current state plus a control $\mathbf{U}$ lead to its subsequent state, in a process that forms a Markov chain.

Several approaches may be used to infer the hidden state given the known controls issued to the robot and the known observed sensor readings. The hidden state is represented as a distribution of probabilities, referred to as a belief.

We have developed exercises that let students implement two key methods: grid localization, in which the belief is represented by a finite histogram of probabilities, and particle filters, in which a set of particles collectively represent hypotheses of the state, and regions of high particle density represent stronger beliefs of the state.

The Assignments

We have developed five assignments: (1) ROS Race, an introduction to ROS by writing a simple reactive control program for a simulated planar robot, (2) grid localization in simulated 1-dimensional world, (3) particle filter localization in the same 1-D world, (4) particle filter localization of (x, y, θ) in a simulated planar world, and (5) particle filter localization with a physical robot in an environment matching the simulated planar world.

ROS Race

The first programming assignment in the course was designed to get the students acquainted with ROS, which includes device drivers, libraries, visualization tools, and integrated simulation environments. A robot control program written for ROS may control either a simulated robot in a simulated world, or a real robot in the real world.

Students were given a race track in Stage (a planar simulated robot world) with a virtual Pioneer robot. Students could access two ROS topics: a motor control topic and a laser range-finder topic. Subscribing to the laser topic provided sensor data from the simulated robot, and publishing to the motor topic moved the robot. Students were required to design a reactive control program that would drive the robot through the race track.

1-D Robot World

For the second and third programming assignments, students were given a simulated 1-dimensional world (Figure 1) modeled after material in (Thrun, Burgard, and Fox 2005). A robot that could travel only left and right was placed in a corridor with three identical doors. Based on motion and a door sensor, the goal was to localize within the corridor.

Because simulation in ROS does not directly introduce real-world error, two nodes were written to add noise to the simulation. A motor node sits between the students' control program and the underlying command-velocity node. This node restricts movements to allow only horizontal movement. Further, it adds noise to the students' velocity command by selecting an actual velocity from a Gaussion distribution centered at the requested velocity.

A "bad sensor" node was designed to give a binary output ($door$, $wall$) based on the underlying simulated laser scanner. Using a sum of a subset of laser scans, the node determines the robot's actual relationship to doors and walls. Then the node adds randomness, as illustrated in Figure 2.

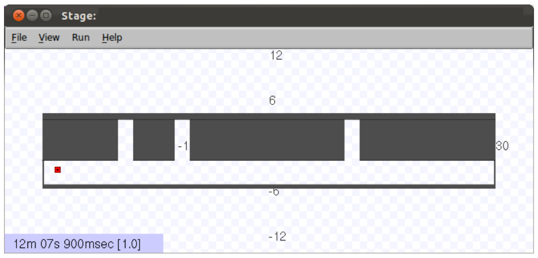
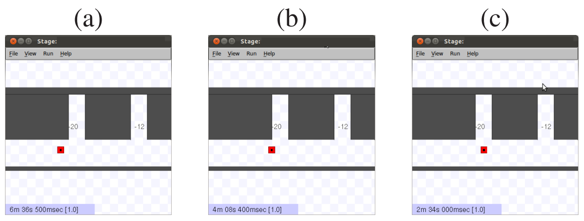


Figure 1: ROS Stage simulation showing robot at the left of a one-dimension world. The robot moves horizontally along the corridor, sensing either a wall or a door.



$P(door) = 0.1$ $P(door) = 0.47$ $P(door) = 0.9$

Figure 2: Probabilistic operation of the door sensor. When robot is located at a wall (a), the sensor reports $door$ with probability 0.1. When located at the edge of a door (b), the sensor reports $door$ with probability 0.47. When located in the middle of a door (c), the sensor reports $door$ with probability 0.9.

1-D Grid Localization

Assignment 2 was designed around this 1-D world. The students were to implement the Grid Localization algorithm described on page 238 of (Thrun, Burgard, and Fox 2005). They were instructed to move the robot at 4m/s and represent the 60m-long corridor as 600 grid cells (each representing 10cm). In this case, the initial probability of each grid cell is $\frac{1}{600}$, representing the robot's initial belief state of maximum uncertainty. Figure 3 illustrates a student's solution to the grid localization assignment.

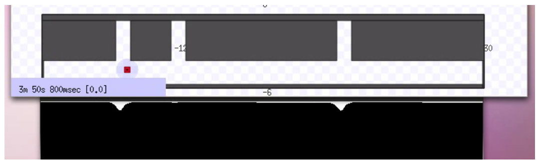


Figure 3: Student solution using grid-based localization. The robot started at the left edge of the world and has traveled into the first doorway. It has seen enough of the wall such that the probability of it being located at the second doorway is small. It is equally likely that the robot is located at the first door or the third door. *Courtesy Michael McGuinness.*

1-D Particle Filter Localization

Particle Filter Localization (also known as Monte Carlo Localization) represents belief state using a particle distribution. PFL has similar properties to Grid Localization, but can be made to run in real time for detailed multi-dimensional worlds. In Assignment 3, students replaced their 1-D grid localization code with PFL.

Planar Particle Filter Localizaton

For Assignment 4, we gave students a Stage-simulated planar world for robot localization. In implementing PFL, students built beam models for the simulated laser, and developed low variance and effective particle sampling methods.

Real World Particle Filter Localization

In the final assignment, students were given a reference localization solution from Assignment 4, and tasked with running it on a physical Bilibot robot operating in an actual corridor. This brought the theoretical localization techniques into concrete practice.

Discussion

The ROS Race was successful in introducing students to ROS. The subsequent assignments were all highly programming-intensive, requiring students to devote significant time to be successful.

The assignments provided a strong basis for demonstrating the essential theory. Students confronted many considerations necessary for creating effective practical solutions, including choices regarding particle counts, resampling methods, and computational efficiency.

Most students worked diligently to create whole solutions from relatively minimal starter code. Others found the design and coding challenge to be too steep. We will consider providing partially working starter code in future semesters.

References

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. MIT Press.

Thrun, S. 2002. Probabilistic robotics. *Commun. ACM* 45:52–57.

Thrun, S. 2010. Toward robotic cars. *Commun. ACM* 53:99–106.