

CS 1567

Intermediate Programming and System Design Using a Mobile Robot

Lab 2: Simple Control and Navigation

Introduction

The purpose of this lab is to transform you into experienced mobile robot programmers by writing position relative control programs that you will be using throughout the semester. The lab is divided into two parts:

- In Part 1 you will add two new methods to the *RobotController* class: Turn(<tenths of degrees>) and Go(<tenths of inches>).
- In Part 2 you will implement a general way-point navigator program that will move the robot through a course that consists of a series of range/bearing (read angle/distance) coordinates.

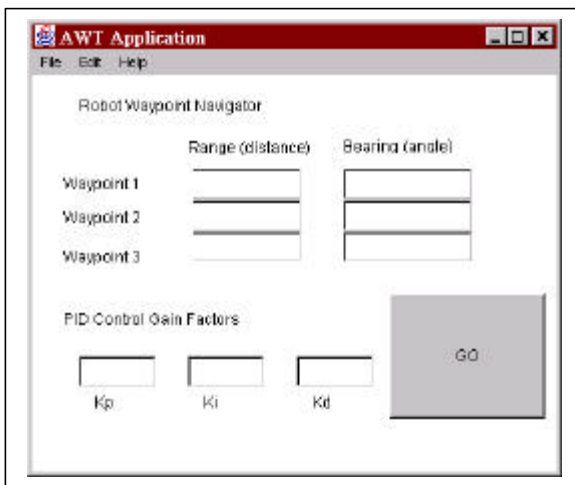
Part 1: Basic positioning functions

Write myTurn(int <tenths-of-degrees>) and add this method to your copy of the RobotController class. When called, the robot will turn <tenths-of-degrees> relative to its current orientation. Degrees are measure counterclockwise from the current orientation. Furthermore, the robot should turn the shortest way, for example, if given Turn(2700) the robot should turn clockwise 90 degrees.

Write GoTheDistance(int <tenths-of-inches>) and add this method to your copy of the RobotController class. When called, the robot moves <tenths-of-inches> forward if <tenths-of-inches> is positive, and <tenths-of-inches> backward if it's negative.

Part 2: Waypoint navigation

Implement a new GUI for the robot that looks like this:



Your program should navigate through one to three way-points by first turning “Bearing” number of tenths of degrees and then moving by “Range” number of cm to each way-point.

We will lay out a course in the lab for you and give you range and bearing for each point. Each group will compete against the others on this course: winners get a gold star for now. Two features will be considered: speed and accuracy. Pay close attention to the notes below.

Notes:

1. Use only the encoders for position feedback information for this lab.
2. To increase speed and accuracy, use PID control and try to compensate for sensor delay. In order to support the user-input gain factors from the GUI, overload your **Turn** and **Go** methods, so that they operate both with and without PID controller gain inputs. For example, implement
 Turn(int angle)
 and
 Turn(int angle int Kp, int Ki, int Kd)
3. Be careful when you tune the gain factors, because your controller may become unstable. Limit the velocity setting to the hardware maximum of plus/minus 300. You may want to test some trial values with a spreadsheet model.
4. Remember that the encoders have a time lag so the robot will always be slightly ahead of where its encoders say it is. Compensate for this.
5. You should seriously consider reconfiguring the acceleration (see robot manual) of your robot to a value of your own choosing.