CIS 32.5 Fall 2009, Project 3

1 Description

This project involves some simple navigation and optionally using a new kind of sensor.

2 Start some navigation

1. You will start from a familiar place, with the setup in the world you see when you type:

   `player world3.cfg`

   As before this should pop up a square window labelled **Player/Stage: ./world2.world** which contains a grey dot and some strangely shaped lumps. This is the simulated world in which your robot will operate.

2. You will start from more or less where you finished last time (depending on exactly where you finished) with the file

   `roomba-local2.cc`.

3. You can see the difference if you compile the controller in `local-roomba2.cc` using:

   `./build local-roomba2`

   and then run it using:

   `./local-roomba2`

4. A strange blue area is projected from the robot. This is a laser mounted on the robot scanning the environment.

5. As you can see from the code — this time helpfully included in this document — you can get hold of information from the laser as well as information about where the robot is at every point in time.

6. Now, what I want you to do is to edit the original version of

   `local-roomba2.cc`

   so that the robot makes makes its way from the location $(-6, -6)$ — which is where the unedited config file I gave you has it starting — to the opposite corner of the space, $(6, 6)$.

7. When you are done, save your program as **proj3.cc** and make sure you put your name in the comments.

8. You'll need to submit this to Prof. Parsons after you are done with the project.

3 Comments

1. The code (this time included in these instructions) gives you some ideas as to how to use the information that the robot can now provide on:

   - Its position.
   - What the laser sees.

2. If the laser data seems confusing, ignore it for now. You can just solve this problem using the data on the robot's position.

3. Of course, this position data is still being produced magically. Next week we'll see how to do the necessary computation.

4. If you are happy using the laser data, you might try navigating by just heading straight for $(6, 6)$ and avoiding anything that gets in the way.

```
/**
 * local-roomba2.cc
 *
 * Sample code for a robot that has two front bumpers and a laser scanner,
 * and magically knows where it is. Suitable for use with the roomba.
 *
 * This is local-roomba.cc, modified to use a laser driver.
 *
 * Written by: Simon Parsons
 * Date:       18th October 2009
 *
 **/


#include <iostream>
#include <libplayerc++/playerc++.h>
using namespace PlayerCc;

/**
 * Function headers
 *
 **/

player_pose2d_t readPosition(LocalizeProxy& lp);
void printLaserData(LaserProxy& sp);
void printRobotData(BumperProxy& bp, player_pose2d_t pose);

/**
 * main()
 *
 **/

int main(int argc, char *argv[])
{


  // Variables
  int counter = 0;
  double speed;            // How fast do we want the robot to go forwards?
  double turnrate;         // How fast do we want the robot to turn?
  player_pose2d_t  pose;   // For handling localization data
  player_laser_data laser; // For handling laser data

  // Set up proxies. These are the names we will use to connect to
  // the interface to the robot.
  PlayerClient    robot("localhost");
  BumperProxy     bp(&robot,0);
  Position2dProxy pp(&robot,0);
  LocalizeProxy   lp (&robot, 0);
  LaserProxy      sp (&robot, 0);

  // Allow the program to take charge of the motors (take care now)
  pp.SetMotorEnable(true);
```

```
  // Main control loop
  while(true)
    {
      // Update information from the robot.
      robot.Read();
      // Read new information about position
      pose = readPosition(lp);
      // Print information about the laser. Check the counter first to stop
      // problems on startup
      if(counter > 2){
        printLaserData(sp);
      }

      // Print data on the robot to the terminal
      printRobotData(bp, pose);

      // If either bumper is pressed, stop. Otherwise just go forwards

      if(bp[0] || bp[1]){
        speed= 0;
        turnrate= 0;
      }
      else {
        speed=.1;
        turnrate = 0;
      }

      // What are we doing?
      std::cout << "Speed: " << speed << std::endl;
      std::cout << "Turn rate: " << turnrate << std::endl << std::endl;

      // What does the laser say?
      //std::cout << "laser: " << sp[0] << std::endl;

      // Send the commands to the robot
      pp.SetSpeed(speed, turnrate);
      // Count how many times we do this
      counter++;
    }

} // end of main()


/**
 * readPosition()
 *
 * Read the position of the robot from the localization proxy.
 *
 * The localization proxy gives us a hypothesis, and from that we extract
 * the mean, which is a pose.
 *
 **/

player_pose2d_t readPosition(LocalizeProxy& lp)
```

```
{

  player_localize_hypoth_t hypothesis;
  player_pose2d_t          pose;
  uint32_t                 hCount;

  // Need some messing around to avoid a crash when the proxy is
  // starting up.

  hCount = lp.GetHypothCount();

  if(hCount > 0){
    hypothesis = lp.GetHypoth(0);
    pose       = hypothesis.mean;
  }

  return pose;
} // End of readPosition()

void printLaserData(LaserProxy& sp)
{

  double maxRange, minLeft, minRight, range, bearing;
  int points;

  maxRange  = sp.GetMaxRange();
  minLeft   = sp.MinLeft();
  minRight  = sp.MinRight();
  range     = sp.GetRange(5);
  bearing   = sp.GetBearing(5);
  points    = sp.GetCount();

  //Print out useful laser data
  std::cout << "Laser says..." << std::endl;
  std::cout << "Maximum distance I can see: " << maxRange << std::endl;
  std::cout << "Number of readings I return: " << points << std::endl;
  std::cout << "Closest thing on left: " << minLeft << std::endl;
  std::cout << "Closest thing on right: " << minRight << std::endl;
  std::cout << "Range of a single point: " << range << std::endl;
  std::cout << "Bearing of a single point: " << bearing << std::endl;

  return;
} // End of printLaserData()

/**
 *  printRobotData
 *
 * Print out data on the state of the bumpers and the current location
 * of the robot.
 *
 **/

void printRobotData(BumperProxy& bp, player_pose2d_t pose)
{
```

```
  // Print out what the bumpers tell us:
  std::cout << "Left  bumper: " << bp[0] << std::endl;
  std::cout << "Right bumper: " << bp[1] << std::endl;
  // Can also print the bumpers with:
  //std::cout << bp << std::endl;

  // Print out where we are
  std::cout << "We are at" << std::endl;
  std::cout << "X: " << pose.px << std::endl;
  std::cout << "Y: " << pose.py << std::endl;
  std::cout << "A: " << pose.pa << std::endl;


} // End of printRobotData()
```