CSCI–4190 Introduction to Robotic Algorithms, Spring 2006
Lab 1: out Monday January 30, to be completed by Thursday February 16

Circumnavigating an obstacle

For this lab, you and your lab partner will write a program that makes the robot approach and circumnavigate a rectangular obstacle at a given distance (i.e., clearance).

Here are the steps you must take to complete this lab:

- Sign up for a 45-minute time slot with your lab partner. We will be using a Yahoo calendar to do the scheduling. Go to `http://calendar.yahoo.com` and log in with username `irarpi2006` and password [see WebCT for password]. Then navigate through the calendar, find a spot marked "Available," and change it to include the names of the people in your group.
- Install the `mpro` package, which includes a client library for talking to the robot, along with a simulator that you can use to test your programs and several utilities. You can find a link to `mpro` from the Lab 1 web page. Download and extract the package and then follow the instructions in the `INSTALL` and `README` files to get up and running. Post questions or problems on WebCT.
- Download the support code from the Lab 1 web page.
- *Before* coming to the lab, read this handout completely, write a program to solve the task, and make sure it compiles and runs with the simulator!

  Because your time for this lab will be limited, it is important that you come to the lab prepared. Your time in the lab should be spent fine tuning your program, not struggling to get your program working!
- Come to the lab (MRC 345) at the proper time. Kris will check your prepared program by asking you to run it in the simulator (on your laptop), review basic procedures for using the robot, and get you started using the robot. A short time before the end of the time slot (or when you are ready), you will demonstrate your program to Kris.

The task

The task for this lab is as follows:

- The robot will start pointed towards a rectangular obstacle (of unspecified size) in the middle of the room. The orientation of the robot will be (as close as possible to) perpendicular to the side of the box. The distance from the robot to the obstacle is also unspecified.
- The robot should drive forwards and stop at a certain distance from the box. This "offset distance" should be an input to your program. (A good value to test your program with is 0.5 m.)
- The robot should circumnavigate the box with a rectangular path. This rectangular path is larger than the box by twice the offset distance.
- The robot should stop at the point where it has completed a circuit around the box.

We will ask you to demonstrate your program using two different offset distance values (one of our choice).

The approach

- In order to accomplish this task, you will need to use sensor feedback. I suggest using the IR sensors.
- You will find the `mpro_rotate()` procedure useful for this lab, but will probably use the `mpro_set_velocity()` procedure rather than the `mpro_translate()` procedure.

Robot API

The `mpro` package contains a library for connecting to the robot over the network from your laptop or other computer. This same library can connect instead to a simulator which you can run on your laptop. The `README` file in `mpro` gives some basic documentation on getting up and running. You should also look in the `examples` directory for some simple client programs (like you'll be writing). Finally, look at the `mproclient.h` file in the `src` directory, which is the header file you will include to use the library. It contains a fair amount of documentation. Here are some basic descriptions of the functions in the library. They all return a value greater than or equal to zero when successful.

- `int mpro_connect(const char *host = "localhost")`

  Connect to the robot or to the simulator.

- `int mpro_disconnect(void)`

  Disconnect from the robot or simulator.

- `int mpro_reset_sim(void)`

  If connected to the simulator, rest it to the initial simulation state.

- `int mpro_set_velocity(float trans = 0.0, float rot = 0.0)`

  Translational velocity is specified in meters per second, rotational velocity in radians per second. Either may be positive or negative. You can command translational velocities of upto ±0.4 m/s and rotational velocities of upto ±1.05 rad/s (±60 deg/s).

- `int mpro_stop(void)`

  stops the robot immediately. equivalent to setting the velocities to zero.

- `int mpro_translate(float distance)`

  Move the robot in a straight line (approximately) the given distance (given in meters). Positive values move the robot forwards, negative values move the robot backwards.

- `int mpro_rotate(float angle)`

  Rotates the robot in place (approximately) the given angle (in radians). Positive values rotate the robot counterclockwise, negative values rotate the robot clockwise.

- `int mpro_reset_odometry(float x = 0.0, float y = 0.0, float t = 0.0)`

  Resets the robot's position measured from the odometry to the given values, distances specified in meters, angle specified in radians.

- `int mpro_get_odometry(float *x, float *y, float *t)`

  Retrieve the robot's position as measured by the odometry.

- `int mpro_get_all_ir(float ranges[MPRO_NUM_IR])`

  Retrieve all the IR sensor readings which are distances in meters. Sensor 0 is pointing forwards, and the index increases going counterclockwise. There are a total of 16 IR sensors. The maximum range is approximately 0.8 m, the minimum range, about 0.1 m.

- `int mpro_get_ir(uint8_t index, float *range)`

  Retrieve a specific IR sensor reading.

- `int mpro_get_all_sonar(float ranges[MPRO_NUM_SONAR])`

  Retrieve all the SONAR sensor readings (distances in meters). Sensor 0 is pointing forwards, and the index increases going counterclockwise. There are a total of 16 SONAR sensors. The maximum range is approximately 10 m, the minimum range, about 0.5 m.

- `int mpro_get_sonar(uint8_t index, float *range)`

  Retrieve a specific SONAR sensor reading.

- `int mpro_get_all_bump(bool toggles[MPRO_NUM_BUMP])`

  Retrieve all the bump sensor readings (1 on `true` means that the bump sensor has been activated, otherwise the value is 0 or `false`). Sensor 0 is pointing forwards, and the index increases going counterclockwise. There are a total of 16 bump sensors.

- `int mpro_get_bump(uint8_t index, bool *toggle)`

  Retrieve a specific bump sensor reading.

Running a sample program

The support code contains the following sample program (`lab1.cpp`). You can modify this program to solve the task for this lab.

```
// Simple stub code for IRA Lab 1.  This program moves the robot
// forward slowly until the front IR sensor reports a range of 1/4
// meter.

#include <mproclient.h>
#include <stdio.h>

int main(int argc, char **argv)
{
  char *host = "localhost";
  if (argc > 1)
    host = argv[1];

  if (mpro_connect(host) < 0) {
    printf("Can't connect to %s!\n", host);
    return 1;
  }

  // if host is a simulator, reset the simulation
  mpro_reset_sim();

  mpro_set_velocity(0.25, 0); // start moving forward at 0.25 m/sec

  float range;
  do {
    mpro_get_ir(0, &range); // get the range from IR 0
  }
  while (range > 0.25);
  mpro_stop();
  return 0;
}
```

To run this program (from UNIX), you would issue the following commands (shown with the output that they produce on my computer). In Windows, you may need a separate command shell for each command (and would then not need the `&` at the end of the first two commands to put those processes in the background).

```
$ mprosim -c lab1.conf &
[1] 17132
[2006-01-29 23:56:41] Initializing server (type 0x1)...
Successfully initialized server

$ mprogui &
[2] 17136
Connecting to localhost:7331...
[2006-01-29 23:56:45] Connection from 127.0.0.1
success!
localhost:7331 is a simulator: requesting the map...
success!
Loading required GL library /usr/X11R6/lib/libGL.so.1.2

$ ./lab1
```

When you're finished, you can shut down the background processes by:

- Typing `q` in the GUI window
- Executing the (UNIX) command: `killall mprosim`

When you come to the lab, you will run your code from your laptop. You will then specify the hostname of the robot in order to connect to it, instead of the simulator. Kris will give you the details in the lab.