Lab 2 Random Wander, Obstacle Avoidance

Reading: *Introduction to AI Robotics (Sec. 4.3)*

(Demonstration due in class on Thursday)

(Code and Memo due in Angel drop box by midnight on Thursday)

Read this entire lab procedure before coming to lab.

****

Purpose: An essential characteristic of an autonomous robot is the capability to navigate in an environment safely. The purpose of this lab is to develop random wander and obstacle avoidance behaviors for the Traxster II. The design of your program should follow the *subsumption architecture*. Layer 0 of your control architecture will be the *collide* and *run away* behaviors to keep the robot from hitting obstacles. Layer 1 will be the *random wander* behavior which passes a random heading every n seconds to the robot.

Equipment:
Base robot
4 IR sensors
3 sonar on servo turret
ruler
keypad, speech module, LCD display
obstacles

Software:
Microsoft Visual Studio.NET 2008 with C#
Serializer.NET library and firmware
Bluetooth transmitter

****

LAB PROCEDURE

****

Part 1 –Range Sensors

There are four infrared and three sonar sensors on a servo on the Traxster II that you may use for obstacle avoidance. The infrared sensors are connected to the analog I/O on the serializer (0: front, 1: left, 2: right, 3:back). The Sharp GPD120 sensors measure between 1.5" and 12". The sonar are connected to the digital I/O on the serializer (5: left, 6: front, 7: right). The Maxbotix MaxSonar –EZ1 Sonar has a reliable range of 6" to 20". In order to confirm functionality, you should take several measurements on each sensor to correlate the distance displayed with the actual distance to an object. You will need to use this information in order to have consistency between the different measurement devices and acceptable robot performance. You should write your code to account for any discrepancies. The best way to display this information would be to use a data table and perform an error analysis (see Table 1).

Table 1: IR and Sonar Calibration Data

Distance (")	IR	Error	Sonar	Error
1				
3				
5				
7				
9				
11				
13				
14				
15				
16				
17				
18				
19				
20				

Part 2 – Layer 0 - Obstacle Avoidance

Now that you are familiar with the range sensors and how to move the robot, create an obstacle avoidance behavior. The *obstacle avoidance* abstract behavior includes a *collide* and *run away* primitive behaviors. For the *collide* behavior, if the forward facing sonar and/or infrared sensor fall between 3 and 6 inches, the robot should halt the forward drive motor. For

the *run away* behavior, create a polar plot of the 7 sensor readings and use the sum to create a repulsive vector to turn the robot. This data should be displayed in some form along with the robot's current heading on the GUI or LCD. Finally, the robot should turn by this angle and move forward a short distance (6 to 12 inches) away from the obstacle (see Figure 1).

.

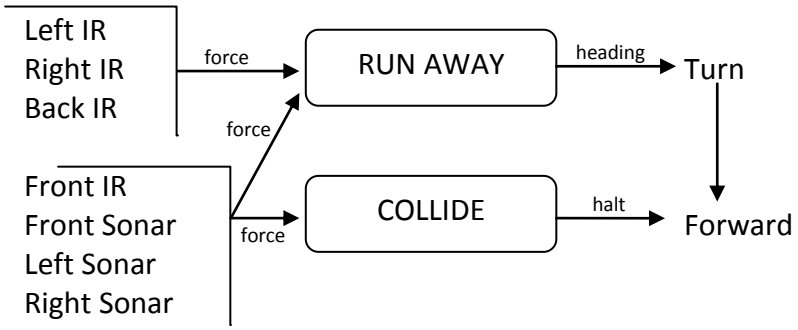


Figure 1: Level 0 – Obstacle Avoidance

Note: You have the option of using the servo on sonar turret to provide more redundant coverage on the robot's front half.

Part 3 – Layer 1 - Random Wander

Create a random wander routine that the robot uses to explore the room. This can be done by generating a random number that represents the robot's heading every n seconds. The current robot heading should be displayed on the LCD or GUI (see Figure 2).

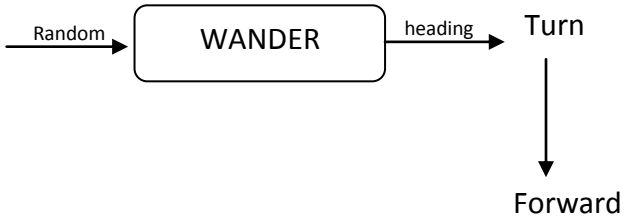
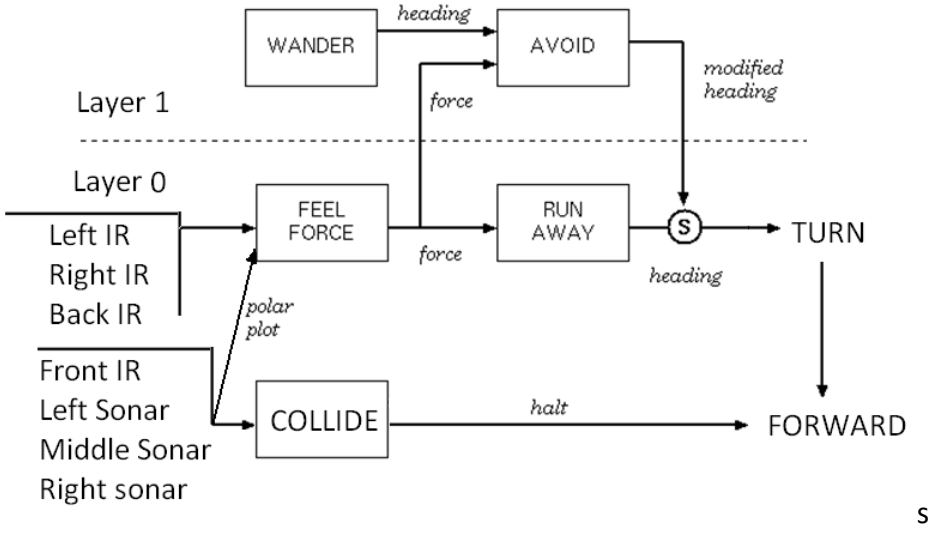


Figure 2: Level 1 – Wander

Part 4 – Subsumption Architecture – Obstacle avoidance and Random wander

Now improve the random wander routine by integrating obstacle avoidance. The robot should wander randomly until an obstacle is encountered. The robot should *RunAway* from the obstacle and continue to wander. The robot's heading from the *Wander* behavior should be modified based upon the force from the range sensors and then turn and move from the obstacle. The *Avoid* module in Layer 1 combines the *FeelForce* vector with the *Wander* vector. The *Avoid* module then subsumes the heading from the Run Away module and replaces it with the modified heading as input to the *Turn* module.



s

Figure 3: Subsumption Architecture - Obstacle Avoidance, Random Wander Example

Your program should provide a method to get the robot 'unstuck' if it approaches any local minima points (i.e. oscillates between two obstacles). Your program should be as modular as possible with multiple subroutines and behaviors that will be integrated in subsequent programs. Devise a method to test and confirm that your program works correctly and present the results in the laboratory memo. Figure 4 provides sample motion for the fully integrated robot behaviors.

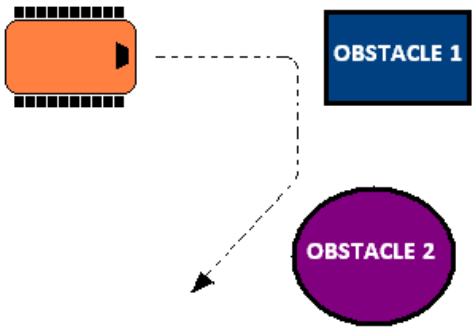


Figure 4: Subsumption Architecture – Sample Robot Motion

Demonstration:

During the demonstration, you will show each layer of the architecture separately. For layer 0, the robot should sit until an obstacle gets close and then it should run away. For layer 1, the robot should turn at a random heading and move forward periodically. Finally, to demonstrate the complete architecture the robot should wander and halt when it "collides" with an obstacle, or modify the heading when it encounters and obstacle and then run away. The robot should give some type of audible and/or visual signal when an obstacle is encountered. The GUI or LCD should show the robot's current heading. Lastly, the LCD or GUI should also display the current state of the robot (waiting, wander or obstacle avoidance). You may use the keypad or GUI to start or stop the robot for the demonstration.

Program:

In subsequent weeks you will reuse this code thus your code should follow proper programming techniques such as detailed commenting and be as modular as possible where behaviors and reactive rules are separate functions. You may start with the *Demo_App.sln* program or create your own from scratch. If you use the sample code, it must be modified to reflect your work and documentation.

Memo:

The following list provides the basic guidelines for writing a technical memorandum.

- ✓ Format
  - Begins with Date, To , From, Subject
  - Font no larger than 12 point font
  - Spacing no larger than double space
  - Written as a paragraph not bulleted list
  - No longer than three pages of text
- ✓ Writing
  - Memo is organized in a logical order
  - Writing is direct, concise and to the point
  - Written in first person from lab partners
  - Correct grammar, no spelling errors
- ✓ Content
  - Starts with a statement of purpose
  - Discusses the strategy or pseudocode for implementing the robot paths (may include a flow chart)
  - Discusses the tests and methods performed
  - States the results including error analysis
  - Shows data tables with error analysis and required plots or graphs
  - Answers all questions posed in the lab procedure
  - Clear statement of conclusions

Questions to Answer in the Memo:

1. What was the general plan you used to implement the random wander and obstacle avoidance behaviors?
2. How did you create a modular program and integrate the two layers into the overall program?
3. Did you use the servo turret to create redundant sensing on the robot's front half.
4. How could you create a smart wander routine to entirely cover a room?
5. What kind of errors did you encounter with the obstacle avoidance behavior?
6. How could you improve the obstacle avoidance behavior?
7. Were there any obstacles that the robot could not detect?
8. Were there any situations when the range sensors did not give you reliable data?
9. How did you keep track of the robot's states in the program?

10. Did the robot encounter any "stuck" situations? How did you account for those?

Grading Rubric:

The lab is worth a total of 30 points and is graded by the following rubric.

Points	Demonstration	Code	Memo
10	Excellent work, the robot performs exactly as required	Properly commented, easy to follow with modular components	Follows all guidelines and answers all questions posed
7.5	Performs most of the functionality with minor failures	Partial comments and/or not modular with objects	Does not answer some questions and/or has spelling, grammatical, content errors
5	Performs some of the functionality but with major failures or parts missing	No comments, not modular, not easy to follow	Multiple grammatical, format, content, spelling errors, questions not answered
0	Meets none of the design specifications or not submitted	Not submitted	Not submitted

Submission Requirements:

You must submit the lab memo and code by midnight on *Thursday*. You must also submit a memo for Lab 1 by midnight on *Thursday*.