IEOR 140: Project 3 Milestone 1

Team 6: MoonSoo Choi & Sherman Siu

**Sherman**: mainly programming coding, task analysis and design

**MoonSoo**: experiments, task analysis and design, reports

**Time spent:** 6 hours

**Project Description:**

We programmed the robot to approach any lights nearby. To detect the location of the light, the robot will use the scanner. While travelling to the light, the robot will continuously measure two parameters: one is distance of the robot away from the light, and two is the angle of robot’s location relative to the light. In order to travel along an appropriate angle, the robot will continuously rotate is scanner left and right to adjust its angle.

Once the robot approaches approximately within 1.5 feet of a beacon, it will turn around, and it will start looking for the light located on the opposite side.

**Most interesting/challenging/difficult part of the project:**

We built a new robot structure for project 3 milestone 1, a different one from project 2, and the overall physical structure was a bit more complicated than we thought. Occasionally our robot would physically break down, unfortunately causing some of the pieces to fall apart... and it did not make us quite happy.

In this milestone, I think it was more of a difficult challenge for us to make the robot turn around and start travelling towards the opposite direction once it gets closer to the light. It was an extra task for us, since travelling towards the light was a continuous process for the robot to keep detecting the light until it gets there, but we had to “*force”* the robot to turn around. We had to think about how we can write our code so that the robot turns around when it’s approximately 1.5 feet away from the light.

Nevertheless, we’re excited for the upcoming obstacle competition among all the robots!

**Task Analysis**

|  |  |  |
| --- | --- | --- |
| **Go to the light\* (milestone 1)** | Detect Obstacles (milestone 2)  Note: Detecting obstacles while travelling | Avoid (milestone 2)  Note: Aborting “obstacle detection” action once the robot spots an obstacle. |
| 1st Class: *goToLight* Class  (*italicized* name is NOT the actual name of the class)   * Takes light intensity and angle (measured by tacho count) as parameters * While the robot is travelling, it updates the sensor’s light intensity value, and value of the angle at maximum light intensity.   Above tasks done by **Racer** class in the milestone | Haven’t done it yet | Haven’t done it yet |
| 2nd Class: *Scanner* class   * Several get methods, such as obtaining the value of the light and the angle. * A method that continuously scans to look for appropriate   Above tasks done by **ScanRecorder** class in the milestone | Scanner class  Probably same as  Milestone 1 | Scanner class  Probably same as  Milestone 1 |
| 3rd Class: a class that delivers  action to the robot   * Merely meant to give “instructions” to the robot, by creating instance variables of above two classes. * All the actions done in the main method   Above tasks done by **Milestone1** class in the milestone | Haven’t done it yet | Haven’t done it yet |

**Experimental work:**

We ran an experiment for this milestone to determine appropriate parameters for the beacon detection scheme and an appropriate location for the robot to turn around and start travelling in opposite direction. In order by locating a robot 1.5 feet away from the beacon and making a 240 degree rotation of a scanner, in order to measure the amount of light intensity.

We made robot’s scanner rotate 240 degrees by first rotating 120 degrees to the left (therefore rotating to 120 degrees), and then by rotating 240 degrees to the right (therefore rotating to -120 degrees)

# The data here actually looks a bit skewed to the left, and it has multiple peaks. Those series of phenomena probably rose from other extraneous factors, such as functionality of the robot. \*

# Nonetheless, the general trend in this graph is that light intensity value becomes lower as motor sensor is farther away from the light.

**Experimental Data results**

**Results Analysis**

\*When we initially placed the robot in front of the light,

I think the location of the light was a bit skewed.

**Overall Control Scheme**

For the angle control for the scanner, we programmed the robot scanner to rotate degrees from the best angle (which is obtained from the *getTargetBearing* method from *ScanRecorder* class). A great advantage of doing so was that robot was able to detect the light relatively quickly, but when the robot straying way too far off the track, having such a narrow scope of angle control didn’t get the robot to get back on track to search for the light. In addition, from the experiment, we can clearly see that 60 is the highest value of light intensity. In order to program the robot to turn 180 degrees when it approaches 1.5 feet within the light, we decided to choose a value slightly lesser than 60. We used *DifferentialPilot’s steer* method in order to make the robot rotate appropriately, and we defined a variable *gain* in order to control the robot.

**Programs:**

Racer.java

Milestone1.java

ScanRecorder.java

JavaDoc’s are in the doc folder in Project 3.