Coverage Efficiency in Autonomous Robots With Emphasis on Simultaneous Localization and Mapping Algorithms

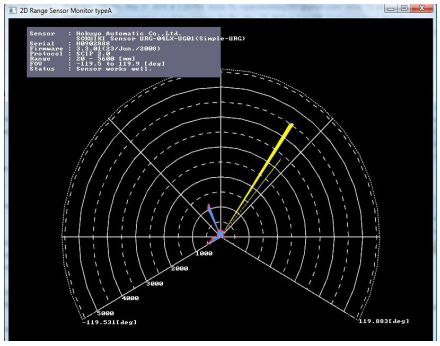
Mo Lu
Computer Systems Lab 2009-2010

Abstract

Today, automated systems have supplemented humans in previously labor-intensive tasks. Automated lawnmowers are an example of these systems, but the currently available technology in automated lawnmowing is inefficient and primitive. This project will propose and implement an alternate method to automated lawnmowing, known as Simultaneous Localization and Mapping.

Background

Current approaches to commercial robotic lawnmowing operate under the idea that if a lawmower is constantly mowing the lawn, then the lawn stays constantly mowed. This is done by a series of random cuts and turns, which if given enough time, theoretically could cover an entire unmowed lawn. This method is horrifically inefficient in terms of time and energy, when backtracking is taken into consideration. This project proposes a different approach to this method: use of mapping techniques to recognize landmarks, avoid obstacles, and navigate an environment This method consists of three parts: 1) Use of a constantly updating laser scanner to recognize obstacles, 2) Creation of obstacle map using the laser data, and 3) Processing that obstacle map for runtime efficiency. Success is determined by how effectively the robot avoids the obstacles, how quickly it runs through the lawn, and how accurate the created obstacle map is.



Rangefinger Sight

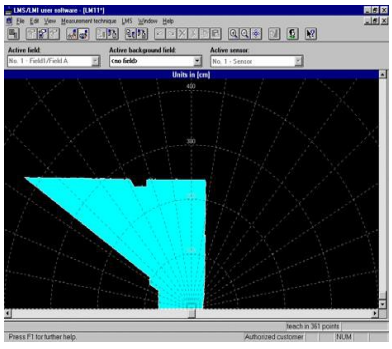
Procedures, Methods, and Results

The entirety of the project code come 4th quarter is split up into two parts. The first part of the code generates a random matrix environment, places obstacles, and runs a representation of the robot through matrix. This part is array/recursive-trace structure based.

The simulation has been fully adapted for use with the Hokuyo Rangefinder. Testing and coding has been focused on making the rangefinder successfully scan the environment that the testing takes place in and utilizing the scanner commands. This involved translating binary inputs into a hexadecimal intermediate step, then finally to an ACSII output. This final output is then given to the GUI which then gives a visual representation of the environment.

The robot can see. This specific part of the total project has been completed.

Simulation Environment



Actual Environment

ACSII Output

0	['1891', '0', '0', '7', '320', '384', '0']
1	['1891', '0', '0', '7', '320', '384', '0']
2	['1891', '0', '0', '7', '320', '384', '0']
3	['1891', '0', '0', '7', '320', '384', '0']
4	['1891', '0', '0', '7', '320', '384', '0']
5	['1891', '0', '0', '7', '320', '384', '0']
6	['1891', '0', '0', '7', '320', '384', '0']
7	['1891', '0', '0', '7', '320', '384', '0']
8	['1891', '0', '0', '7', '320', '384', '0']