Coverage Efficiency in Autonomous Robots With Emphasis on Simultaneous Localization and Mapping Algorithms
TJHSST Senior Research
Computer Systems Lab 2009-2010

Mo Lu

June 14, 2010

Abstract

Coverage efficiency is a major goal of certain autonomous robotic systems. In the field of robotic lawnmowing, coverage efficiency has yet to be fully developed and there are different methods to approach coverage efficiency. The solution this paper covers is uses Simultaneous Localization and Mapping, known as SLAM. Using a laser scanner, SLAM algorithms create a map detailing the obstacles of the enviornment. Once obstacles are mapped, the algorithm process the map, and dictates where the robot can move, where it has moved, and where it currently is in relation to the obstacles. This data will enable the robot to cover the entire lawn.

Keywords: map processing, area efficiency

1 Introduction

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

2 Background

Commercial autonomous lawnmowers today do not have processing systems appropriate for efficient coverage. 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[1]. This is done by a series of random cuts and turns, which if given enough time, theroetically could cover an entire unmowed lawn[1]. Another aspect of this method is the use of "bump-and-go" technology. The system does not recognize the presence of obstacles until it actually hits it, and when it does hit obstacles, it does not store their locations for future use. This method is horrifically innefficient in terms of time and energy, when backtracking is taken into consideration. Random cuts also contain the possibility that a certain section of the lawn will never get mowed. This project proposes a different approach to this method: use of mapping tecniques to recognize landmarks, avoid obstacles, and naviagate an enviornment[4]. 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[2]. Success is determined by how effectively the robot avoids the obstacles and how quickly it runs through the lawn.

3 Development

3.1 Theory

SLAM theory is centered around the mapping process. A laser scanner is mounted on the robot, and pings out laser data in a 180 degree angle. The time it takes for the laser to hit an obstacle determines how far the obstacle is. These values are tracked by the system while the scanner is constantly working, and repeated obstacle values signify an obstacle, which the robot maps in relation to its current position. Once the obstacles are mapped, the robot will be able to process the most viable and efficient route through the lawn, taking into consideration the obstacles, terrain, and boundaries of the lawn. It will also take account power sources and effective runtime. The end result will enable the robot to navigate and mow the lawn.

3.2 Project Work

Before the SLAM algorithims can be implemented into a physical robot, it must first run in a simulation. The final version of the simulation consists of a pre-created matrix based enviornment where the obstacles and terrain have been set. The robot is placed in the environment and keeps track of its position and obstacles, via the use of a coded coordinate system, a scanner mimic which has a 2 space range, and a blank obstacle map. As the robot moves and scans through the enviornment, obstacles are recognized, and the robot begins to build on its own independent matrix enviornment. The output of this mapping process matches the locations of the obstacles in the enviornment, and gives the robot an idea of where it can and cannot move in future mowings. The program is advanced enough to navigate and map vertical, horizontal, diagional, and circular obstacles. It also can recognize the boundaries of an environment. The simulation has been fully adapted for use with the Hokuyo URG-04LX Rangefinder. This has included trans-

lating the 'scanning' methods of the simulation into C++, which the rangefinder software supports, unlike Python. Also, this translation has involved incorporating base code structures from the rangefinder software into the code, most of which covers appropriate command calls for the rangefinder to send out pings at certain rates. The concept behind the rangefinder is nearly the same as the mimic from the simulation; a ping out that measures the time for the ping back, which determines the distance to an obstacle. Current version has been tested for a triangular environment, which incorporates the diagional, vertical, and horizontal obstacle recognition. Also, analysis has been done on the runtime effencieny of the simulation scanning, in order to optimize certain sections and clean up laggy code. The finalized code takes the original inputs given to the URG which is in binary form, and converts that into a hexidecimal intermediary format. This is then translated into an ACSII which is then fed to the GUI. Though the end result is the same as if using binary, the extra translation steps allow the user to cross check for disrepencies between the visualizaion and the inputs.

4 Testing and Analysis

The most general test of the performance of the system is if it mows the lawn. This depends on wether or not it maps the environment accurately. When efficiency is taken into account, three new categories for testing arise:

- Time efficiency
- Coverage precentage
- Backtracking

These testing categories are dependent on obstacle and boundary recognition, obstacle mapping, location tracking, and unmowable terrain recognition. One aspect of the testing is focused on obstacle/boundary recognition and obstacle mapping. This testing is determined by how accurate the obstacle map is when compared to the environment. The simulation itself, specifically the robot movement, has not been tested in a non-matrix based enviornment, but the translated code has been tested in those types of environments to a certain degree. The non-matrix based enviornments (physical environments) has given results, but only the scanning has been tested. Many conditions still must be met for success of this project, if the original goal is to be met. This involves on getting the robot to move without assistance, in order to fully ensure that the navigational aspects of the simulation can be translated into the robot platform. Any future testing will address the processing aspect of the program, with sucess determined by coverage and time efficiency.

5 Results

In the simulation, the robot is correctly placed in the enviornment, and obstacles are generated. See Fig. 1. Red represents

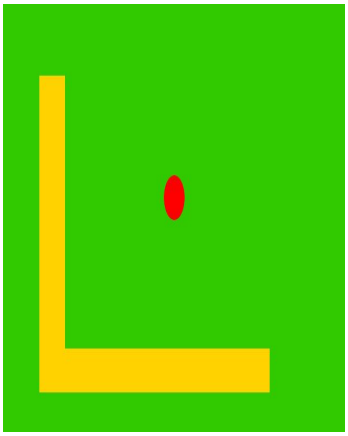


Figure 1: Environment

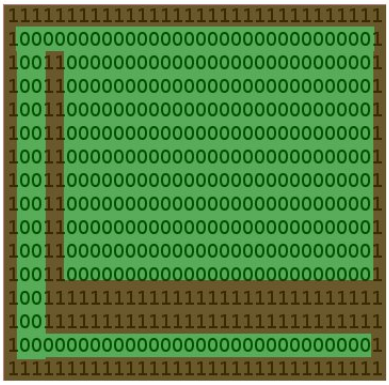


Figure 2: Modified Environment

the lawnmower, yellow represents the boundaries.
Mapping algorithims print out a matrix-based map. See Fig. 2. [1] represents an unmoveable zone, and [0] represents moveable zones.
Current inputs include diagional, vertical, horizontal, and circular obstacles. See Fig. 3/4/5. Obstacles represented by figure 5 have been tested with the rangefinder. See Fig. 6

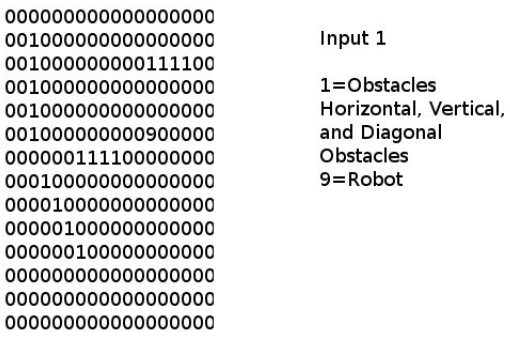


Figure 3: Input Environment: Diagional, Vertical, and Horizontal

6 Discussion

Before the SLAM algorithms can be implemented into a physical robot, it must first run in a simulation. The current version of the simulation consists of a pre-created matrix based environment where the obstacles and terrain have been set. The robot is placed in the environment and keeps track of its position and obstacles, via the use of a coded coordinate system and a scanner mimic. The

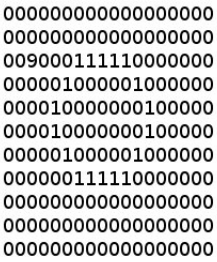
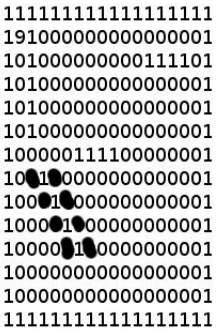


Figure 4: Input Environment: Circle

Figure 6: Output: Triangular/Simulation



Output 1

Notice the border broundaies and the deadzone around the diagional obstacle.

Figure 5: Output: Diagionals

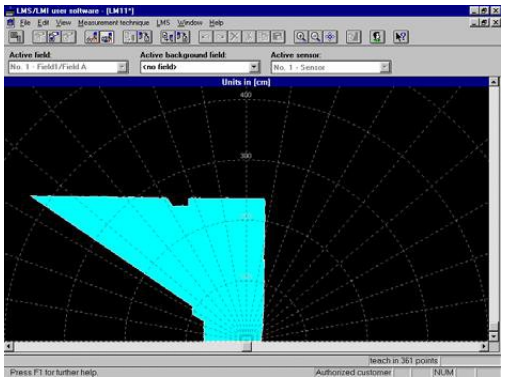


Figure 7: Output: Triangular/Rangefinder

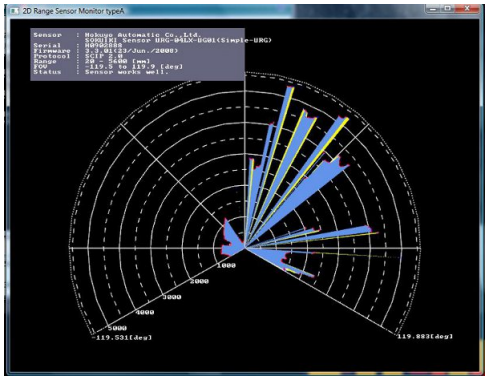


Figure 8: Scanner Vision 1

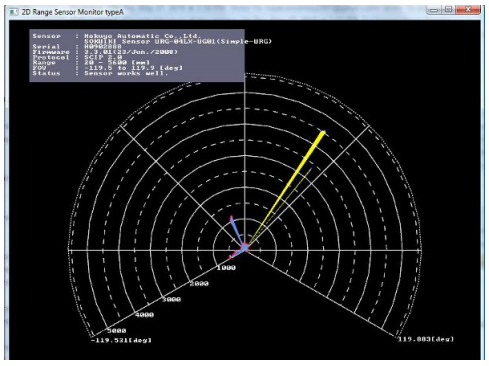


Figure 9: Scanner Vision 2

```
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']
```

Figure 10: Tranlated ACSII Output, 8 scan

robot moves and scans through the environment so long as obstacles are a certain distance away, and the environment map does not equal the obstacle map. Obstacles are recognized, and the robot begins to create its own independent matrix environment. Since the output of this mapping process matches the locations of the obstacles in the environment, it can be concluded that the scanning and obstacle recognition works for certain obstacles. That, along with the robot's ability to keep track of its position gives all the nessacary data to begin optimization algorithims. The simuilation has been tested for non-matrix based environments (graphic-based). However, only the scanning portion of the code has been tested in this environment. Because non-matrix based enviornments cannoot have a coordinate system, the robot must process its location based off odemetry (wheel movement calculations) and its last known position. In order for this to be tested, the robot must move on its own. One problem that needs to be address in the current code is the tendency to re-scan already known obstacle locations. Future versions will need to reflect more realistic conditions such as terrain types and powersources.

7 Conclusion

The final version of the program gives all the nessacary data for optmization processing to begin. Also, the program is advanced enough to be translated for use in a real environment and has been tested as such. While the robot can see and process an physical envi-

ronment, movements still must be considered and tested before the project can be considered fully complete.

References

[1] Husqvarna, "Husqvarna Automower", http://www.automower.us, 2009.

[2] Sren Riisgaard and Morten Rufus Blas, "SLAM for Dummies", pp. 1-44, 2003.

[3] Ian Schworer, "Navigation and Control of an Autonomous Vehicle", pp. 1-84, 2005.

[4] Dustin Bates and Evan Dill, "The Ohio University Autonomous Lawnmower", pp. 1-21, 2009.

[5] SICK Sensor Intelligence, "LMSIBS Configuration Software and Operating Instructions", pp. 1-71, 2010.

[6] Se, Lowe, Little, "Mobile Robot Localization and Mapping using Scale-Invariant Visual Landmarks", 2004

8 Acknowledgments

Special thanks to Captain Randy Latimer for his guidance throughout the project. Thanks to Mr. DC for providing the Hokuyo Laser Rangefinder, along with Mr. Kemp. Credit goes to the team: Jeff Hobson (Systems Lab,) Andrew Palmer (Robotics Lab,) Matthew Chan (Energy Systems Lab,) and Victor Yu (Energy Systems Lab.) Please feel free to contact with any follow up questions at ml3qf@virginia.edu.