Jurnal Teknologi

Full Paper

MOBILE ROBOT PATH OPTIMIZATION ALGORITHM USING VECTOR CALCULUS AND MAPPING OF 2 DIMENSIONAL SPACE

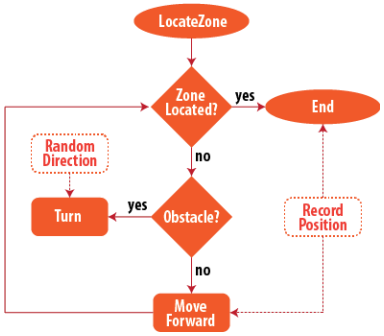
Ammar Zahari*, Amelia Ritahani Ismail, Recky Desia

Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia (IIUM), P.O. Box 10, 50728 Kuala Lumpur, Malaysia

*Corresponding author
ammar.zahari@live.com

Article history
Received
15 May 2015
Received in revised form
1 July 2015
Accepted
11 August 2015

Graphical Abstract



Abstract

This research explores path integration in mobile robot navigation and path optimization technique using vector calculus. A simulated robot in a simulated environment is used to test the algorithm that is to be developed. The simulated robot is equipped with a sonar sensor and several infrared sensors on its chassis. Mobile robot navigation in an unknown environment is very crucial as It not only has to be concerned about reaching its destination but also to avoid obstacles that may be in the way. This algorithm can effectively allow a mobile robot to navigate an unknown environment without collision into obstacles.

Keywords: Robotics, mapping, artificial intelligence

1.0 INTRODUCTION

Autonomous robotic has long been a field of interest among researchers. Programmable robot with the ability to multi-task and navigate treacherous terrains where it would be otherwise dangerous for a human to traverse are highly desirable. Vast amount of accurate sensors that are equipped by the robot are capable of calculating and processing information in fractions of the time it would take a human to do. This has become an ambitious area of research in the last few decades [1].

Research on mobile robot navigation and mapping has become increasingly popular these days [2] [3] [4] [5] [6]. This paper aims to develop a simple algorithm or Finite State Machine (FSM) that can navigate an unknown environment and accomplish a simple task. The fundamental aspect of a decentralized mobile robot system is the ability to make decisions without communicating commands with a control center. We will explore the behavior of a mobile robot to navigate with precision using its own sensors and information. The need for a decentralized mobile robot system to identify its location is critical in navigating an unknown environment. We proposed an algorithm based on simple vector calculus and path integration. To achieve this goal the mobile robot must be able to construct a map based on its environment, to localize itself in it and to move from one point to another by simulating foraging behavior of insects. The scope of this paper is limited to a single simulated robot in a simulated world. Physical robots or multiple robots are not used in this paper.

Robotics is a machine that senses and converts into action using computers. Typically robots have sensors such as vision, force, tactile as internal states [7]. Actions that are performed are provided by arms, grippers, wheels and sometimes legs [7]. Artificial Intelligence is a vast field of study. In its essence it is related to incorporating problem solving skills with the though process that is present in complex biological organisms [8]. Mobile robotics as the name implies are robotics that is mobile as opposed to static robots.

Ants and termites are social insects, they cooperate with each other to achieve certain tasks for example foraging and navigation. There are many species of

ants that use many different methods for navigation. One of those methods is landmark-based navigation where the ants would use the environment around them as a beacon or check-points during navigation [9]. Ants also use a form of path integration, a record of its distance and direction traveled is updated in an accumulator [10].

Path integration is a simple navigation method. It continuously computes its current location based on its past location and trajectory, this means that to return to its original location, instead of taking a direct route, the path will be retraced from its entire trajectory [11].

2.0 RELATED WORK

Cong *et al*. in [3] applies fuzzy logical in their self-navigating robots. A fuzzy logic system consists of fuzzy variable membership functions, a rule base, fuzzy reasoning and defuzzification. Cong demonstrated the advantages of this approach by applying it to the wheels of a robot. Their demonstration successfully navigated an area although the map was incomplete and barely recognizable.

Luo *et al*. in [4] applies sensor-based navigation with grid map representation for their robots to navigate unknown environments. They demonstrated a biologically inspired neural network model algorithms and real-time map building. Although their method is very successful, they equipped their robot with expensive equipment including laser radar or LIDAR.

Zhou *et al*. in [6] applies an RSSI-based localization of robots. RSSI is an indication of the power level being received by the antenna which is related to distance between sender and receiver. They use RSSI in their paper to determine the distance between sender and receiver and use that as a beacon or landmark for navigation. Although their work was successful RSSI is a wireless technology that would greatly be affected by interference.

Mariappan *et al*. in [5] uses optical tracking devices in their robot to navigate. This is very useful for certain tasks, but it requires a lot of processing power and very expensive. Their robot uses an omni-wheel system whereby the robot is able to move in all direction. They demonstrated its navigation capabilities and it was very successfully.

3.0 METHODOLOGY

3.1 Experimental Setup

3.1.1 Stage

Stage is a robot simulator [12]. It provides a virtual world populated by mobile robots and sensors, along with various objects for the robots to sense and manipulate. In this paper, the virtual world is populated by a single robot with 2 zones, one zone represents the nest and the other represents a resource. The world is within a small and simple 2-dimensional maze. Controller for the robot is implemented in C++.

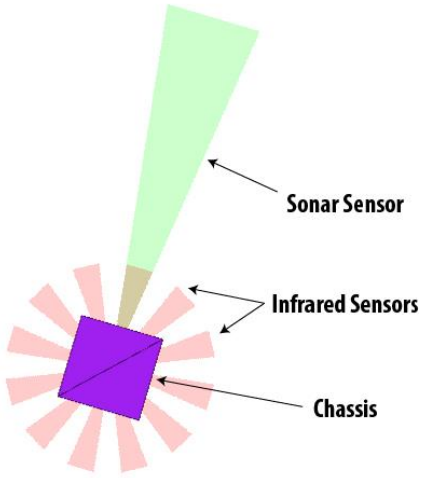


Figure 1 Simulated robot sensor placement

There are 12 simulated infrared sensors that surround the robot at 30 degree intervals as in Figure 1. The infrared sensor is used as a proximity bumper while the sonar sensor is used as a distance measurement tool. The infrared sensor works by turning on an infrared LED, the light beam then bounces of an object then return back to a photo-resistor. The amount of resistance that is detected by the photo-resistor determines the distance from the robot to any obstacle. Values from the photo-resistor will change the voltage going through it. This will allow a value to be read. A Sonar sensor is placed at the front of the robot to measure distances between itself and any obstacles in-front of it, the sonar sensor works by sending out an ultrasonic wave, this wave will bounce of an obstacle then is received by an ultrasonic microphone. The time it takes for the wave to come back will be measured, with the known value of the speed of sound; the distance can easily be measured.

3.1.2 Data Acquisition

The nest zone has a controller which is also implemented in C++. When the nest receives a resource item, it will calculate the time it takes for each resource item it takes. This value will then be used to calculate the efficiency of the robot by calculating its resources per minute value.

$$R = \frac{60}{T_2 - T_1}$$

Equation 1 Resources per minute

Where, R is resources per minute, T_1 is the time-stamp of when the previous resource item was received and T_2 is the current resource item. The value R will then be sent to a graphing software to provide visual information.

For every movement made by the robot, its bearing, speed and the obstacle distance in-front of it is sent to

the mapping system to be drawn as a map. If there are no obstacles, only the bearing and speed data is sent.

3.1.3 Data Processing and Visualization

SocketGrapher is an application that is developed in Java using the JavaFX library. It reads datagrams from UDP sockets and displays them in a line graph. SocketGrapher accepts 2 numbered data separated by a comma for its, X-Axis and Y-Axis. SocketGrapher is capable of exporting its data to a CSV (Comma Separated Values) file.

AbizMapper is an application that is used to provide mapping visualization. It is developed in Java using LibGDX library. It reads datagrams from UDP sockets and processes the data to draw a map on the screen. AbizMapper accepts data such as bearings and ranging data that corresponds to obstacles that need to be drawn on screen.

3.2 Algorithm Design

3.2.1 Navigation

The movement of the simulated robot is random. Initially the robot will move forward until it encounters an obstacle, then it will change its direction until there are no obstacles in-front of it. This will continue until the robot locates either the nest zone or the resource zone. Whichever zone the robot locates first, it will start recording its movement while finding the other zone. The simulated GPS coordinates of the robot along with its bearing will be recorded and inserted into a vector of values as in Algorithm 1.

Algorithm 1 Navigation Algorithm

```
1:  procedure LocateZone (Z, R) > Z, Zone to be located and R, Recording flag
2:      currZone = GetCurrentZone ()
3:      while currZone ≠ Z do
4:          if ObstacleFront() = false then
5:              MoveForward()
6:              if R = true then
7:                  RecordPosition()
8:              end if
9:          else
10:             direction ← Random()
11:             Turn(direction)
12:         end if
13:     end while
14:     return CurrentPosition() > Zone found at position
15: end procedure
```

3.2.2 First Stage Optimization

The first stage of optimization removes redundant positions from the recorded positions. When the robot moves, it will detect obstacles that are somewhat near to each other then it will keep turning to find its way out. These turns will flood the position vector with garbage data. Thus the first stage of optimization will remove them. These garbage data will be found by calculating the distance between one position and the next.

$$D = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Equation 2 Distance equation

Where D is the distance, X_1 and Y_1 is the first point and X_2 and Y_2 is the second. The distance equation is applied to each pair of position and whichever distance is less than that of the minimum allowed distance will be removed from the vector of positions as in Algorithm 2.

Algorithm 2 Stage 1 Optimization Algorithm

```
1:  Procedure OptimizeStage1(P) > Vector of positions
2:      size ← NumElements(P)
3:      loop from i ← 0 to size
4:          curr ← P[i] > Current point
5:          next ← P[i + 1] > Next point
6:          min ← GetMinimumDistance()
7:          If Distance(curr, next) less then min then
8:              remove next from P
9:          end if
10:     end loop
11:     return P
12: end procedure
```

3.2.3 Second Stage Optimization

The second stage of optimization is done every time the robot reaches its destination. The path to its next destination is recalculated and an attempt for optimization is made. Each iteration of the algorithm, a point in the path is marked for optimization. When the robot reaches this point, it will skip the marked point in an attempt to go straight to the next point in the path. If the robot succeeds the point will be marked as skip able and will be skipped for all subsequent iterations. Figure 2 shows the skipped path (Original Path) while the resultant path is the path that is made by using simple vector calculus.

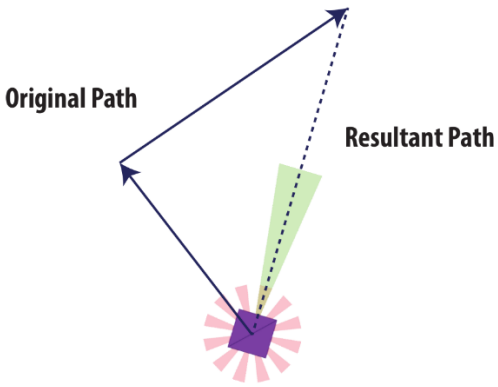


Figure 2 Vector integration

Algorithm 3 Stage 2 Optimization Algorithm

```
1:  procedure OptimizeStage2(S) > Vector of point status
2:      size ← NumElements(S)
3:      loop from i ← 0 to size
4:          if S[i] = skip then
5:              S[i] ← skippable
6:          end if
7:          if S[i] = untested then
8:              S[i] ← skip
9:          end if
10:     end loop
11:     return S
12: end procedure
```

4.0 ANALYSIS OF RESULTS

From the data that is sent by the simulated robot, SocketGrapher software will plot its data. SocketGrapher plots its resources collected per minute versus time. Figure 3 at the 7th minute mark the speed at which the robot collects resources shows 0.25 resources per minute. At the 15th minute mark the robot begins to run its second stage of optimization algorithm. The optimization algorithm improves the robot's resource collection rate to 0.55 resources per minute at the 47th minute mark. This is approximately 120% improvement before any optimization is made.

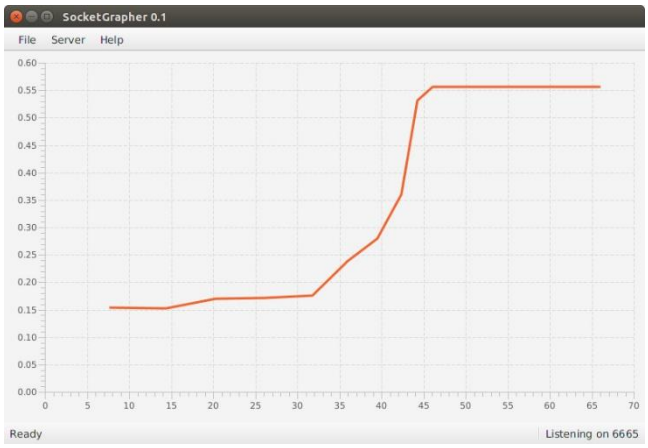


Figure 3 Graph for resources per minute versus time

Figure 4 is the Stage simulator where it consists of a world with a small 14m x 12m cave and populated by a single 80cm diameter robot. Figure 5 shows the SIRD Control Center mapping system where it receives ranging data from the simulator and draws the resulting map. The map that is drawn has some visible "holes" in the walls at multiple locations at certain areas. The walls that are drawn appear to have multiple layers to them where the same wall is drawn multiple times.

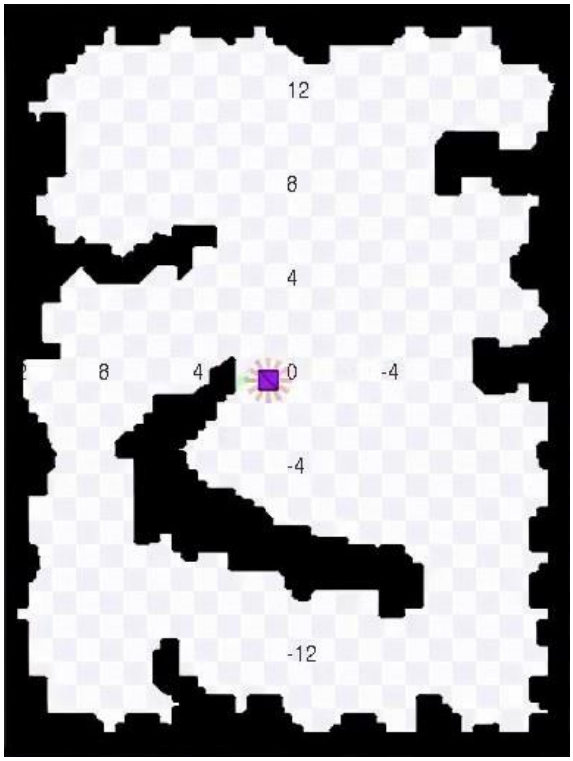


Figure 4 Simulated map

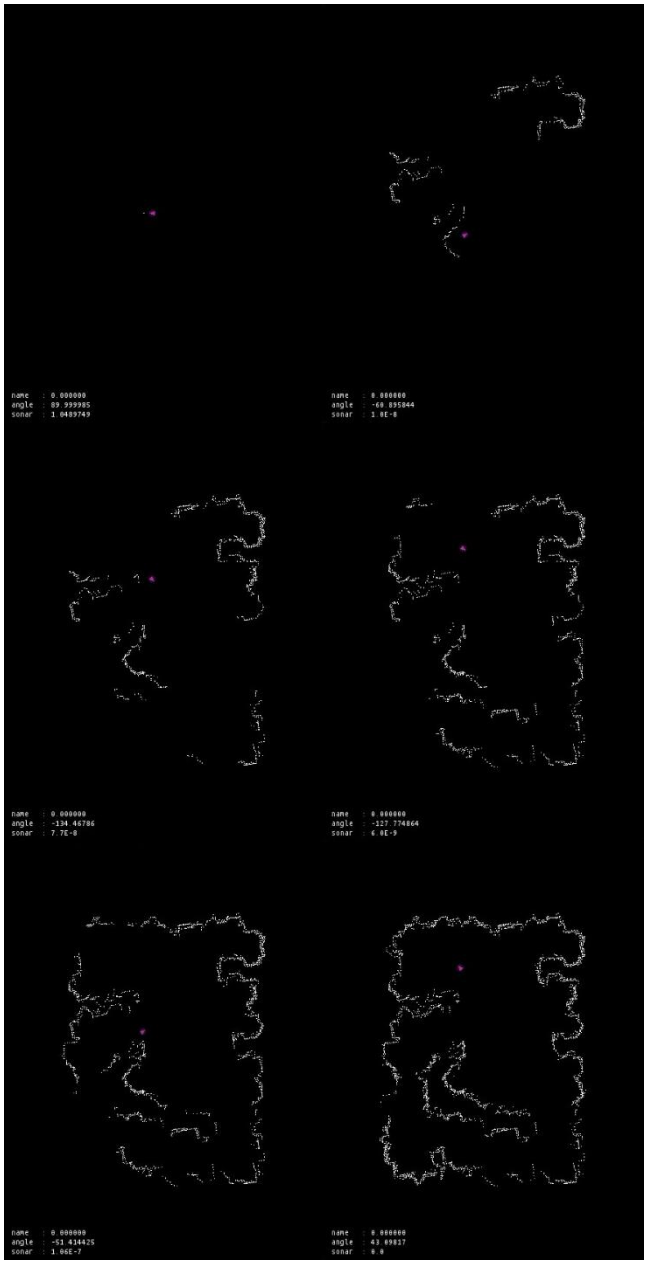


Figure 5 Results for the mapping algorithm tests from top left to bottom right

5.0 DISCUSSION

This research sets out to develop a path navigation and optimization algorithm as well as a 2 dimensional mapping system. There were 2 separate simulations that was done for the optimization and the mapping respectively.

The first simulation concerns that of the optimization of the pathway that is taken by the robot to reach its destination. It is observed that new paths can only be created from an already known path. New paths could not be created from unknown data. This may lead to a local maximum that wouldn't lead to an absolute optimized pathway.

The second simulation concerns that of the mapping system. The robot takes a considerable amount of time to map a relatively small area. Since the robot moves randomly it maps the same area several times and this leads to multiple layers of the same wall or obstacle.

6.0 FUTURE WORK

Further work can be done on this research to improve its methodology, this includes but not limited to:

- Physical robots to observe emergent behaviors that may not be apparent in simulations.
- Multiple robots (swarm robotics) to improve data collection time and further optimize pathways.

Acknowledgement

This research is funded by the Ministry of Higher Education under the Exploratory Research Grant Scheme (ERGS): ERGS 13- 017-0050.

References

[1] M. A. Hossain and I. Ferdous. 2014. Autonomous Robot Path Planning in Dynamic Environment Using a New Optimization Technique Inspired by Bacterial Foraging Technique. In *International Conference on Electrical Information and Communication Technology (EICT)*. 1-6.

[2] K. Sugawara, M. Sano, I Yoshihara, K. Abe, and T. Watanabe. 1999. Foraging Behaviour of Multi-Robot System and Emergence of Swarm Intelligence. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 3: 257-262.

[3] Xiaohong Cong, Hui Ning, and Zhibin Miao. 2007. A Fuzzy Logical Application in a Robot Self Navigation. In *Industrial Electronics and Applications, 2007. ICIEA 2007. 2nd IEEE Conference on*. 2905-2907.

[4] Chaomin Luo, Jiyong Gao, Xinde Li, Hongwei Mo, and Qimi Jiang. 2014. Sensor-based Autonomous Robot Navigation Under Unknown Environments with Grid Map Representation. In *Swarm Intelligence (SIS), 2014 IEEE Symposium on*. 1-7.

[5] M. Mariappan, Choo Chee Wee, K. Vellian, and Chow Kai Weng. 2009. A Navigation Methodology of an Holonomic Mobile Robot Using Optical Tracking Device (OTD). In *TENCON 2009-2009 IEEE Region 10 Conference*. 1-6.

[6] Nan Zhou, Xiaoguang Zhao, and Min Tan. 2013. RSSI-based Mobile Robot Navigation in Grid-Pattern Wireless Sensor Network. In *Chinese Automation Congress (CAC), 2013*. 497-501.

[7] Michael Brady. 1985. Artificial Intelligence and Robotics. *Artificial Intelligence*. 26(1): 79-121.

[8] E. S. Brunette, R.C. Flemmer, and C. L. Flemmer. 2009. A Review of Artificial Intelligence. In *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*. 385-392.

[9] T. S. Collett. 1996. Insect Navigation En Route to the Goal: Multiple Strategies for tThe Use of Landmarks. *The Journal of Experimental Biology*. 199(1): 227-235.

[10] Matthew Collett, Thomas S. Collett, and RÃ¼diger Wehner. 1999. Calibration of Vector Navigation in Desert Ants. *Current Biology*. 9(18): 1031-S1.

[11] L. Delahoche, C. Pégard, E.-M. Mouaddib, and P. Vasseur, *1998.* Incremental Map Building For Mobile Robot Navigation In An Indoor Environment. In *Robotics and Automation,*

Proceedings. 1998 IEEE International Conference on. 3: 2560-2565.

[12] Richard Vaughan. 2008. Massively Multiple Robot Simulations in Stage. In *Swarm Intelligence 2(2-4)*. Springer. 189-208.