International Journal of Engineering & Technology, *7 (3.33) (2018) 28-32*

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A Study of Autonomous Navigation of a Robot Model based on SLAM, ROS, and Kinect

Asilbek Ganiev[1], Kang-Hee Lee[2]*

[1]Department of Digital Media, Soongsil University, Seoul, Republic of Korea, 06978
[2]Department of Digital Media, Soongsil University, Seoul, Republic of Korea, 06978
**Corresponding author E-mail :kanghee.lee@ssu.ac.kr*

Abstract

In this paper, we used a robot operating system (ROS) that is designed to work with mobile robots. ROS provides us with simultaneous localization and mapping of the environment, and here it is used to autonomously navigate a mobile robot simulator between specified points. Also, when the mobile robot automatically navigates between the starting point and the target point, it bypasses obstacles; and if necessary, sets a new path of the route to reach the goal point.

Keywords: *ROS; SLAM; Autonomous Navigation; Robot Simulation.*

1. Introduction

In this paper, we propose work that uses simultaneous localization and mapping (SLAM) to create maps of an area, and then use this map to reach a certain (user-defined) place quickly, and bypass obstacles that are on the robot path.In this work, we made three scenarios to navigate the robot model in the simulator. First: autonomously navigating the robot model from point A to point B. Second: autonomously navigating the robot model between multiple goal points (from point A to B, B to C, and C to D). Third: autonomously navigating the robot model to the multiple goal points by avoiding obstacles.

2. Related Research

This section reviews related researches such as SLAM, ROS, and Turtlebot.

2.1. Simultaneous Localization and Mapping

In robotics mapping, simultaneous localization and mapping (SLAM) is a process in which a mobile robot can map an unknown environment, and at the same time, use this map to compute its location. Mapping is the problem of integrating the information collected from the robot's sensors. In this process, the robot answers the question: "What does the environment look like?" The main aspects in the construction of maps are the representation of environmental data, and the interpretation of sensor data. On the other hand, localization is the problem of determining the location of the robot on the map. At the same time, the robot answers the question "Where am I?" [1].

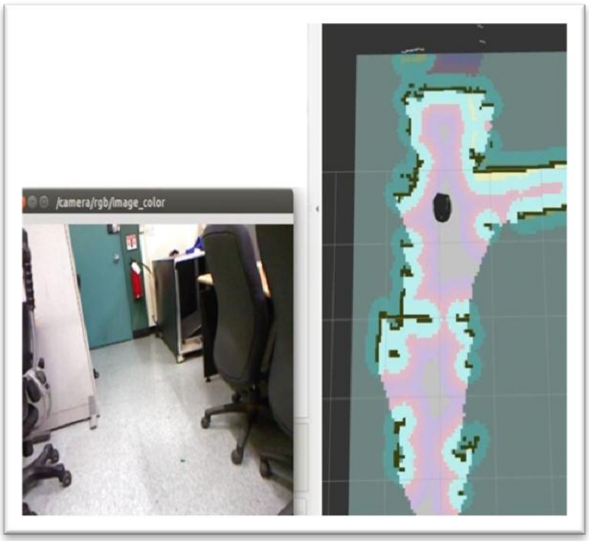

Fig. 1: Example of Simultaneous Localization and Mapping and the research room experimented in this paper

2.2. Robotic Operating System

ROS architecture is based on a graph where the data is processed at nodes that can transmit and receive messages among themselves. The library is focused on Unix-like systems (Ubuntu Linux is included in the "supported" list, while other options, such as Fedora, Mac OS X, and Microsoft Windows are considered "experimental"). ROS has two main "sides": the operating system side ros, and ros-pkg, the set of supported package users (organized into sets that are called stacks) that implement various robotics functions: SLAM, planning, perception, modeling, etc. ROS is available in accord with the terms of the BSD license, and with open source. ROS is free to use, for research and commercial purposes. Packages of ros-pkg are distributed under various open source licenses [2].

2.3. Turtlebot

Turtlebot 2 is a personal robot with an affordable price and with open source software [3][4]. Turtlebot 2 can create a robot that can drive around the user's house, in 3D (refer to Figure 2).

Fig. 2: TurtleBot2, personal robot.

2.4. Costmap

The complexity of constructing a path in a task for a robot is due to the inertness of the robot and its nonlinear dimensions; therefore, most path planning algorithms for robots use costmap. Costmap determines the "cost" of the passage of the robot at one or another point of the map, where a higher cost means the trajectory passing through this pointis "more expensive" [5]. For the robot to avoid collision, the footprint of the robot should never intersect a red cell, while the center point of the robot should never cross a blue cell [6]. Refer to Figure 3.

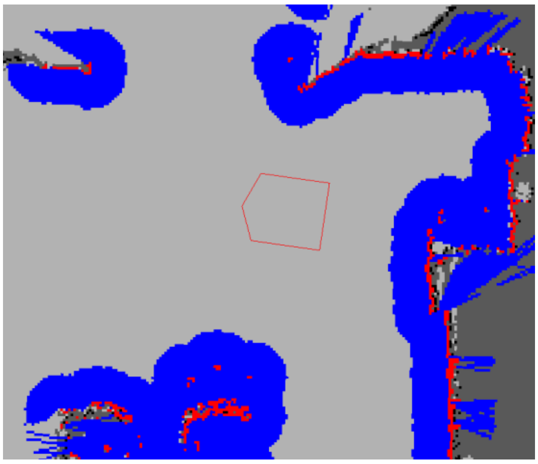

Fig. 3: Example of a costmap.

3. Navigation Stack and SLAM

Detailed submission guidelines can be found on the journal web pages. All authors are responsible for understanding these guidelines before submitting their manuscript.

3.1. Navigation Stack

The Navigation Stack is fairly simple on a conceptual level. It takes information from odometry and sensor streams, and outputs velocity commands to send to a mobile base [7]. However, use of the Navigation Stack on an arbitrary robot is a bit more complicated, but it is possible to identify the main parts - the "purpose" of the route is processed by the global planner, which transfers control to the local planner, which already sends specific values of robot speeds.

3.2. A* Algorithm

In computer science, A* is a computer algorithm that is widely used in path-finding and graph traversal, the process of plotting an efficiently directed path between multiple points, called nodes. Due to its performance and accuracy, it enjoys widespread use. However, in practical travel-routing systems, it is generally outperformed by algorithms that can pre-process the graph toattain better performance [8], although other work has found A* to be superior to other approaches.
A* is a variant of Dijkstra's algorithm that is commonly used in games. A* assigns a weight to each open node equal to the weight of the edge to that node, plus the approximate distance between that node and the finish. This approximate distance is found by the heuristic, and represents a minimum possible distance between that node and the end. Once an initial path is found, this allows it to eliminate longer paths. If there is a path of length x between the start and finish, and the minimum distance between a node and the finish is greater than x, that node need not be examined [9][10].

4. Experiments

Implementation of the algorithm occurs in two stages. The first includes implementation of the algorithm and its testing in software simulation. The next stage is launching the algorithm on the real robot, and adjusting it. Operations of the algorithm in the simulator and on a real robot use a common ROS platform. The modeling of the algorithm is performed in three steps: first, making a map of the environment, second, path planning, and third, path following.

4.1. Map Building

In this experiment we used SLAM, which technology allows a robot with a laser scanning device to simultaneously draw a map, and determine its location on the map. To draw a map and control the simulator robot, we used Kinect device, RTAB Map, Rviz, and ros stage simulator.
SLAM is used in mobile autonomous vehicles and robots to build maps in unknown space, or to update the map in a well-known space, with simultaneous monitoring of the current location, and the distance traveled.
Our experiments were conducted at Soongsil University Media Robotics Lab. Figure 4shows the process of map building by using the SLAM method.

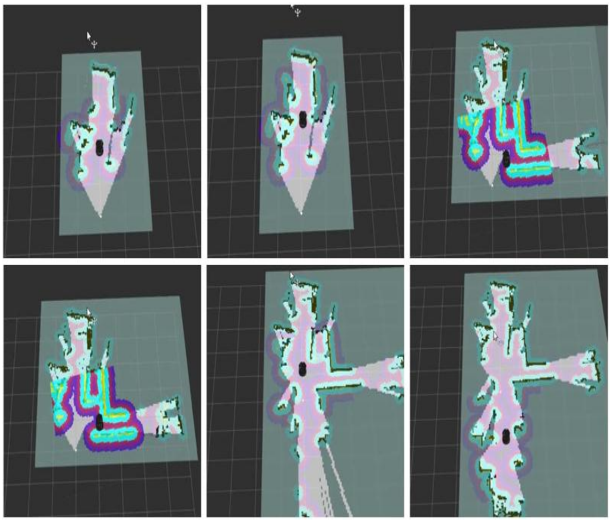

Fig. 3: .Process of map-building by using the SLAM method.

Figure 5 shows the map we get after building a map of the area by using the SLAM method. This map shows broken black borders.

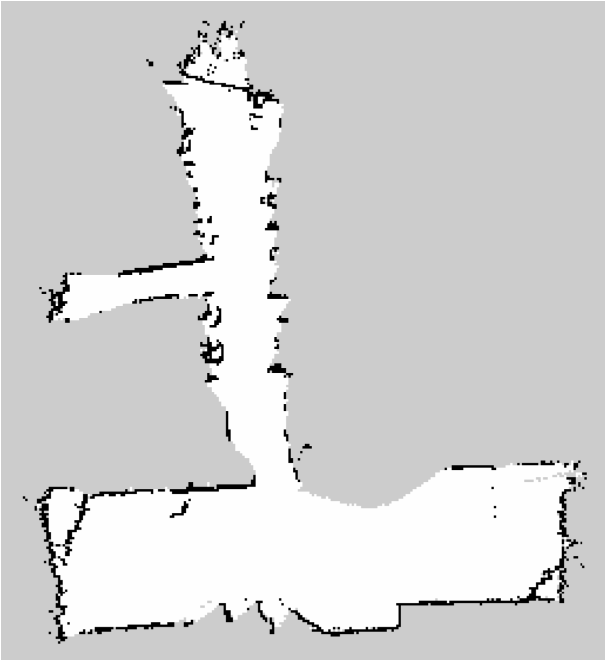

Fig. 5: Map from SLAM mapping. Scanned with Kinect XBOX 360

These non-connected parts appear during scanning by kinect device. The kinect device camera view angle is 57° in the horizontal, and 43° in the vertical. For that reason, these parts in the map do not have borders.

To solve this problem and connect borders (black lines) in the map, we made an algorithm. Table 1shows the RBG data of the map, which has three different color numbers 0 – Black (Borders); 204 – Grey (Unknown area); 254 – White (suitable area for movement of the robot).

Table 1: Map RGB data

0	1	2	3	4	5	6	7	8	9	10	11	12
1	204	204	204	204	204	204	204	204	204	204	204	204
2	204	204	0	254	254	254	254	254	0	0	0	0
3	204	254	254	254	254	254	254	254	254	254	254	254
4	0	254	254	254	254	254	254	254	254	254	254	254
5	204	254	254	254	254	254	254	254	254	254	254	254
6	0	254	254	254	254	254	254	254	0	254	254	254
7	204	254	254	0	0	0	0	0	204	204	204	204
8	204	204	204	204	204	204	204	204	204	204	204	204
9	204	204	204	204	204	204	204	204	204	204	204	204

This is a simple code for use in Matlab, but by using this method, we can accurately connect each broken line. For connection points in the map, we use the width and height of the map. This algorithm will check confidences between two lines of width and height. For example if the 1st column has A and the 2nd column has B, then change A to C, or if the 1st column has B and the 2nd column has A, then change A to C.

Figure 6 shows the results of the modified map with connected broken lines:

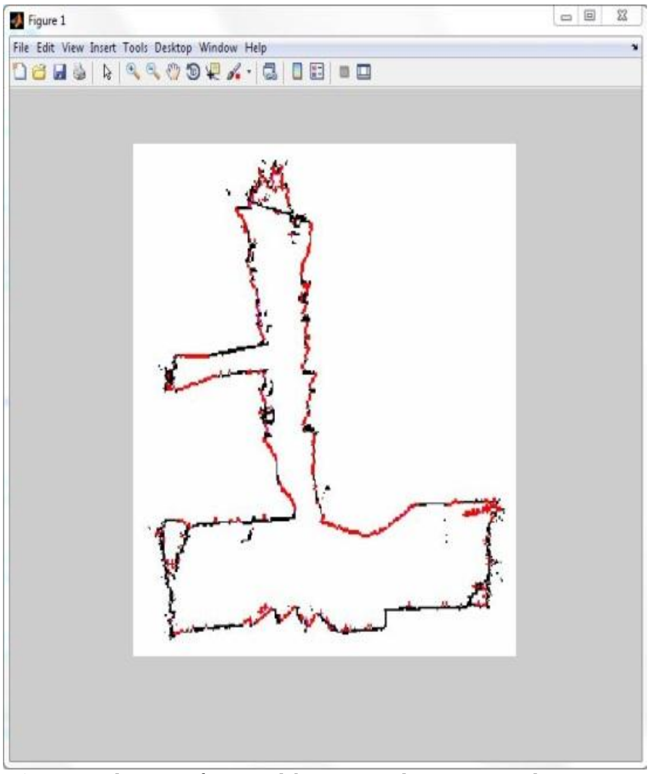

Fig. 6: Result images from Matlab. Connected non-connected parts

4.2. Path Planning

To plan paths, we have to know the coordinates of the map. Then, by using those coordinates, we can send messages to navigate the robot model. The lower part of Figure 7 shows the coordinates of the map and the robot model coordinates. Figure 8 shows the console window to send a destination to the robot in each scenario.

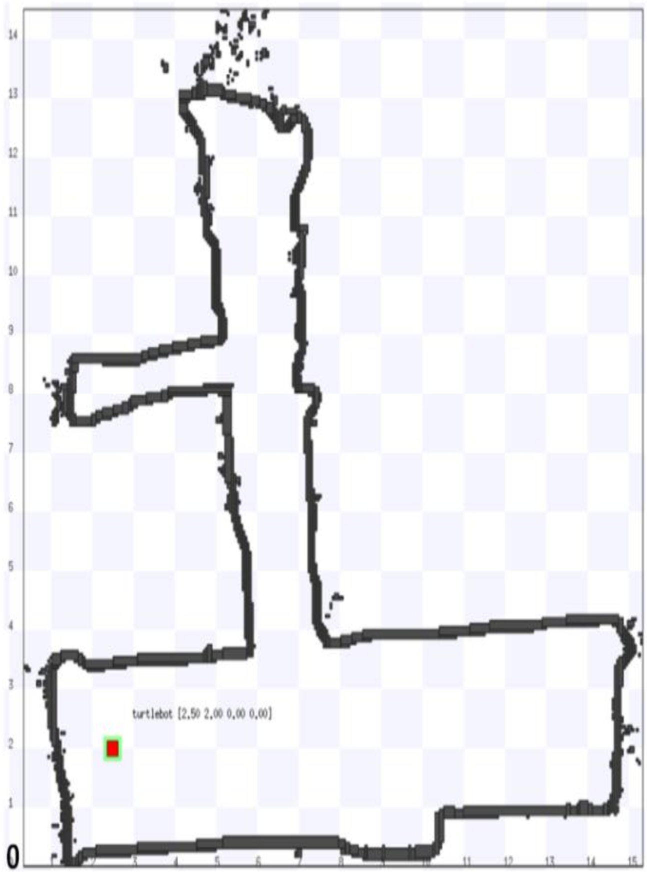

Fig. 7: Coordinates of the map

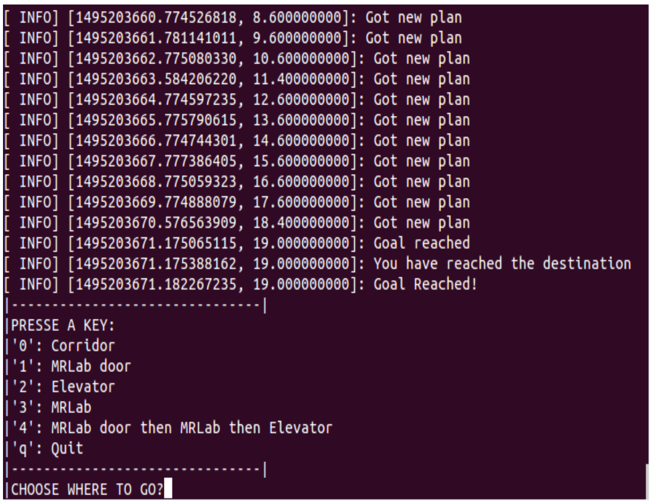

Fig. 8: Console window to send a destination to the robot.

In this work, we made three scenarios for autonomous navigation of the robot model:

First scenario: The robot should autonomously reach a goal point (from point A to point B). Refer to Figure 9.

Second scenario: The robot should autonomously reach multiple goal points (From point A to B, then from B to C, and from C to D). Refer to Figure 10.

Third scenario: The robot should autonomously reach multiple goal points by avoiding obstacles in the robot's path. Refer to Figure 11.

Table 2: Coordinates of each location.

Corridor	X = 2.30
	Y = 1.75
MRLab Door	X = 6.85
	Y = 1.75
MRLab	X = 6.00
	Y = 10.50
Elevator	X = 13.30
	Y = 2.70

4.2.1. First Scenario

The robot should autonomously reach a goal point (from point A to point B). The fist executed console commands are as follows:

catkin_make - Building and using catkin packages in a workspace.

relaunchasilbekmap_navigation_stage_MRLab.launch-Open the RViz, stage simulator and console, to send the destination point to the robot.

Figure 9 shows that the robot autonomously follows the path form

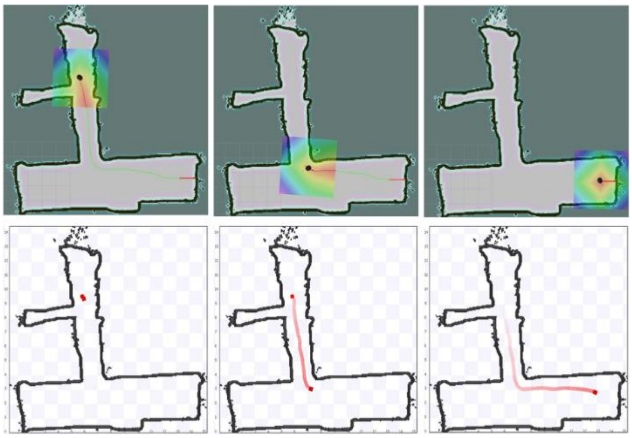
Fig. 9: Scenario 1: Path following from MR Lab to elevator

4.2.2. Second Scenario

Robot should autonomously reach multiple goal points (From point A to B, then from B to C, and from C to D). Figure 10 shows the results for autonomous navigation of the robot to multiple goal points.

Table 3: Goal point coordinates in the map (A, B, C, and D).

Point A	X = 2.30
	Y = 1.75
Point B	X = 6.85
	Y = 1.75
Point C	X = 6.00
	Y = 10.50
Point D	X = 13.30
	Y = 2.70

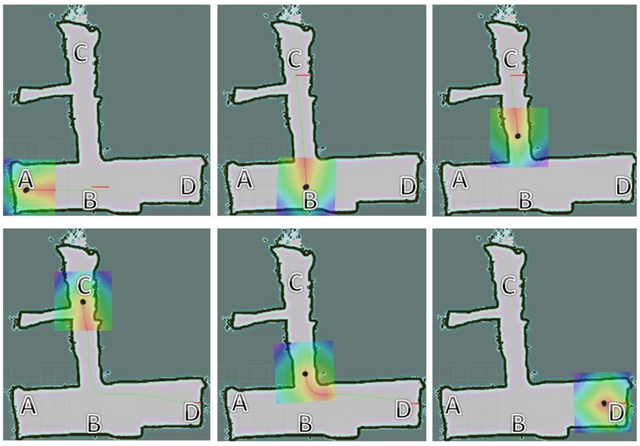

Fig.10: Scenario 2: Autonomous navigation to multiple goal points(from point A to B, B to C, and C to D).

4.2.3. Third Scenario

Robot should autonomously reach multiple goal points (from point A to B, and B to C) by avoiding obstacles. Figure 11 shows maps with obstacles in the robot's path.

Table 4: Goal point coordinates in the map (A, B, and C).

Point A	X = 2.30
	Y = 1.75
Point B	X = 6.85
	Y = 1.75
Point C	X = 6.00
	Y = 10.50

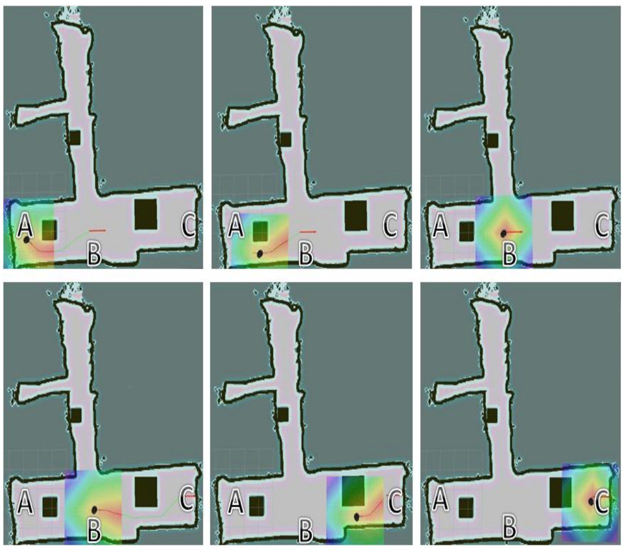

Fig.11: Scenario 3: Autonomous navigation by avoiding obstacles(from point A to B, and B to C).

5. Conclusion

In this paper, we studied the ROS operating system for the development of robots, studied in detail many of its aspects, and studied the ROS navigation stack and the development of rostopic system modules.

The main works of this paper are to make a map of the needed area by the SLAM method, then plan paths to enable autonomous navigation of the robot model. In this work, we made three scenarios to navigate the robot model in the simulator. First: autonomously navigating the robot model from point A to point B. Second: autonomously navigating the robot model between multiple goal points (from point A to B, B to C, and C to D). Third: autonomously navigating the robot model to the multiple goal points by avoiding the obstacles. Also, during our work, we found that after making a map of the needed area, some unmapped and unnecessary spaces did not have borders; to solve this problem, we used Matlab with our algorithm to make borders in the broken parts of the map.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2017R1D1A1B05036028).

References

[1] D. Stoyanov, A. Davison, G. Z. Yang Simultaneous Stereoscope Localization and Soft-Tissue Mapping for Minimal Invasive Surgery. MICCAI. Lecture Notes in Computer Science(2010), 347–354.

[2] A. Martinez, E. Fernandez, Learning Ros for Robotics Programming, (2013), 8-10.

[3] Turtlebot, http://www.turtlebot.com/about/

[4] Turtlebot, https://www.clearpathrobotics.com/turtlebot-2-open-source-robot/

[5] M. Becker,D. Dantas, C. Meirelles , MACEDO, W. Perdigao, Obstacle Avoidance Procedure for Mobile Robots,ABCM Symposium Series in Mechatronics, Volume 2. 1 ed.(2006) Sao Paulo - v. 2, 250-257.

[6] 2D Costmap, http://wiki.ros.org/costmap_2d

[7] P. Agarwal, G.D. Tipaldi, L.Spinello, C.Stachniss, W.BurgardRobotics and Automation (ICRA), IEEE International Conference (2013) on, 62-67,

[8] D. Delling, P.Sanders, D. Schultes, Engineering route planning algorithms. Algorithmics of Large and Complex Networks: Design, Analysis, and Simulation. Springer. (2009), 117–139.

[9] A* Algorithm.http://www.raywenderlich.com/4946/introduction-to-a-pathfinding

[10] ROS global_planner, phttp://wiki.ros.org/global_planner#Examples_of_Different_Parameterizations.