ROS 기반 불안정한 환경 맵 빌딩 테스트

박태환O, 이강희*
O숭실대학교 미디어학과,
*숭실대학교 미디어학과
e-mail: taewhan@soongsil.ac.krO, kanghee.lee@ssu.ac.kr*

ROS-based Uncertain Environment Map-Builing Test

Tae-Whan Park*, Kang-Hee LeeO
ODept. of Digital Media, Soongsil University,
*Dept. of Digital Media, Soongsil University

● 요 약 ●

주로 맵 빌딩 테스트는 불안정한 환경이 아닌 안정된 환경을 조성한 후에 이루어진다. 본 논문에서는 인위적인 안정된 환경이 아닌 불안정한 환경에서 맵 빌딩을 테스트한다. 맵 빌딩 테스트를 위하여 터틀봇3 버거를 사용한다. 터틀봇3의 라이더 센서를 이용하여 맵 빌딩을 진행한다. 터틀봇3는 라즈베리파이로 제어되며 맵 빌딩과 터틀봇3 제어를 위해서는 ROS를 사용한다. 터틀봇3는 우분투와 ROS가 설치된 컴퓨터와 네트워크 통신을 하며 맵 빌딩을 한다. 불안정한 환경에서 맵빌딩이 동작 및 오동작하는 모습을 확인하였으며, 향후 이를 보완하기 위한 방향을 제시한다.

키워드: 맵 빌딩(map building), 터틀봇3(Turtlebot3),
라즈베리파이(Raspberry pi), ROS(Robot Operating System)

I. Introduction

로보티즈에서 제공하는 터틀봇3 e-manual등을 보면 터틀봇3는 맵 빌딩을 할 때 장애물은 거의 없고 안정적인 벽면으로 둘러진 공간에서 진행한다 [1]. 본 논문에서는 맵 빌딩을 위해 만들어진 인위적인 장소가 아니라 좀 더 자연스럽고 불안정한 환경에서 맵 빌딩을 테스트하였다.

1. Related works

1.1 터틀봇3

터틀봇3은 전 세계에서 유명한 교육 및 연구용 오픈소스 로봇이다. 손쉽게 조립, 유지보수, 교체 및 변형이 가능하며 비교적 저렴한 가격의 로봇이다 [2]. 오픈소스 하드웨어이기 때문에 회로도, PCB 커버, 3D 도면 등 연구 개발에 필요한 대부분의 정보가 공개되어있다. 터틀봇3은 ROS를 이용하여 제어된다.

1.2 SLAM

SLAM은 Simultaneous Localization and Mapping의 약자이다. SLAM은 임의의 공간에서 GPS같은 외부 위치 확인 시스템(External Positioning System)에 의존하지 않고 이동하면서 주변을 탐색할 수 있는 로봇을 이용하여 그 공간의 지도를 작성하며 현재 위치를 추정한다 [3][4]. SLAM은 gmapping, cartographer, hector, karto 등 다양한 방식들이 있다. 본 논문에서는 hector 방식을 사용한다.

SLAM을 통해 작성한 지도는 그림1에서처럼 3가지 영역으로 나뉜다. 흰색은 로봇이 이동 가능한 자유 영역(free area)이다. 검은색은 이동 불가능한 점유 영역(occupied area)이다. 회색은 아직 확인되지 않은 미지의 영역(unknown area)이다.

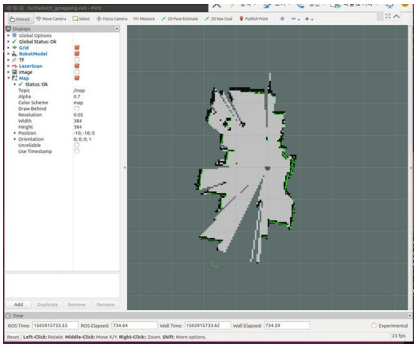

Fig. 1. Map

II. ROS-based Uncertain Environment Map Builing Test

본 실험에서는 Ubuntu 16.04와 ROS의 kinetic버전이 설치된 컴퓨터와 ROS kinetic 버전이 설치된 터틀봇3 버거를 사용한다. 맵 빌딩을 위해 SLAM을 진행할 때 hector 알고리즘을 사용한다. SLAM을 진행하여 맵을 작성할 때는 로보티즈에서 제공하는 터틀봇3 e-manual을 참고한다. 터틀봇3 네비게이션은 Gaitech edu에서 제공하는 패키지와 강의를 참고한다 [5].

1. SLAM

SLAM 방식 중에서 hector을 사용한다. 터미널 창에서 그림2와 같이 SLAM 노드를 실행시킨다. SLAM 노드를 실행시킨 후에는 터틀봇3를 원격제어하며 지도를 작성한다.

2. Map-based Navigation

SLAM으로 작성한 맵을 사용하여 네비게이션을 한다. 터미널 창에서 rostopic echo /amcl_pose를 입력 후 실행하여 주요 위치에서의 로봇의 맵에서의 좌표 값 들을 기록하였다.

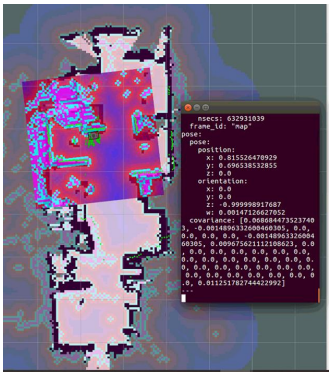

Fig. 2. 맵과 위치를 디스플레이한 모습Map and Position(설명 정확히 영어랑 한글 같이)

좌표들을 기록한 후에 gaitech_edu에서 제공하는 패키지를 이용하였다 [4].

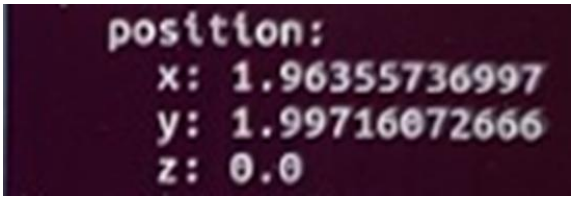

Fig. 3. Positon

지도에서의 로봇의 위치를 알기 위해 amcl_pose 토픽을 사용한다. Navigation 중에 rostopic echo /amcl_pose를 입력하면 그림3과 같이 터미널 창에 출력된다. 실험에서는 좌표에서 소수점 2번째 자리까지 기록하여 사용한다. (실제 연구실사진)

Fig. 4. Tables(실제 맵과 빌딩된 맵 비교)

실험은 연구실에서 진행한다. 잘 정돈된 상태가 아닌 바닥에 물건들이 있고 의자들이 있는 상태에서 실험하였다. 그림4와 같이 연구실에 있는 총 8개의 책상을 목적지로 하여 실험하였다. ROS를 로보티즈에서 제공하는 네비게이션 튜토리얼을 이용하고 터틀봇3을 직접 이동시키면서 각 책상에서의 좌표들을 amcl_pose 토픽을 이용하여 기록하였다.

그림5는 SLAM을 통하여 작성한 지도를 RVIZ창으로 시각화한 모습이다. 검정 색 부분은 장애물이며 연한 보라색 부분은 로봇이 이동가능한 공간이며 진한 보라색 공간은 로봇이 navigation을 통해 현재 인식하고 있는 부분이다.

그림5와 같이 책상에서의 x,y 값을 정리하였다. 1,2,3,4,5,6,7,8을 입력하여 미리 설정한 8가지 장소의 좌표 중 한 장소로 이동시킬 수 있다

표1은 로봇을 책상1~책상8까지 이동시켰을 때 목표 좌표와 실제로 도착한 좌표를 표로 정리하였다.

```
/** declare the coordinates of interest **/
double xDesk1 = 2.53;
double yDesk1 = -0.01;

double xDesk2 = 2.26 ;
double yDesk2 = 0.29;

double xDesk3 = 1.16 ;
double yDesk3 = -0.01;

double xDesk4 = 0.66 ;
double yDesk4 = 0.53;

double xDesk5 = -0.94;
double yDesk5 = -0.22;

double xDesk6 = -1.10;
double yDesk6 = 0.75;

double xDesk7 = -1.14;
double yDesk7 = 0.54;

double xDesk8 = -3.96;
double yDesk8 = 0.77;
```

Fig. 5. Desks Position

Table 1.

Desk	목표 좌표	도착 좌표
Desk1	x=2.53, y=-0.01	x=2.39, y=0.00
Desk2	x=2.26, y=0.29	x=2.26, y=0.31
Desk3	x=1.16, y=-0.01	x=1.19, y=0.01
Desk4	x=0.66, y=0.53	x=0.63, y=0.54
Desk5	x=-0.94, y=-0.22	x=-0.94, y=-0.23
Desk6	x=-1.10, y=0.75	x=-1.10, y=0.70
Desk7	x=-1.41, y=0.54	x=-1.10, y=0.56
Desk8	x=-3.36, y=0.77	x=-3.93, y=0.79

특징 그림 분석

2.1 책상1

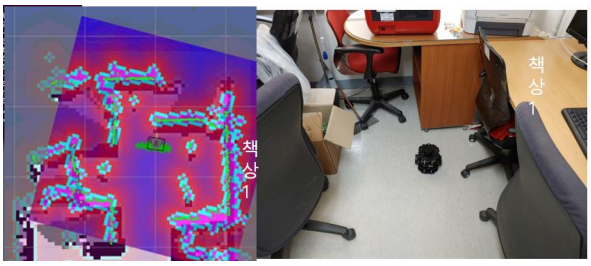

Fig. 6. Desk 1

그림6은 책상1에서의 터틀봇3의 실제 모습과 RVIZ창에서의 터틀봇3의 모습이다. 목표 좌표가 x=2.53, y=-0.01이였고 도착 좌표는 x=2.39, y=0.00이여서 x축으로는 0.14, y축으로는 0.01만큼의 차이가 있었다. 그림7을 보면 책상과 의자를 잘 인식하였다. 의자의 경우 지지대가 얇은 기둥이기 때문에 RVIZ창에서는 마치 점처럼 표시된다.

2.2 책상2

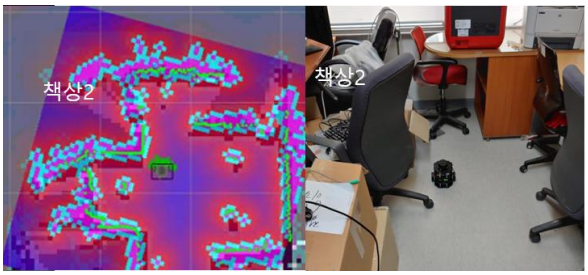

Fig. 7. Desk 2

그림7은 책상2에서의 터틀봇3의 모습이다. 목표 좌표는 x=2.26, y=0.29 였으며 도착 좌표는 x=2.26, y=0.31이다. x축은 정확히 일치하였으며 y축은 0.02만큼의 오차가 있다. 책상1에서와 같이 책상과 의자와 벽의 인식은 잘 되었다.

2.3 책상8

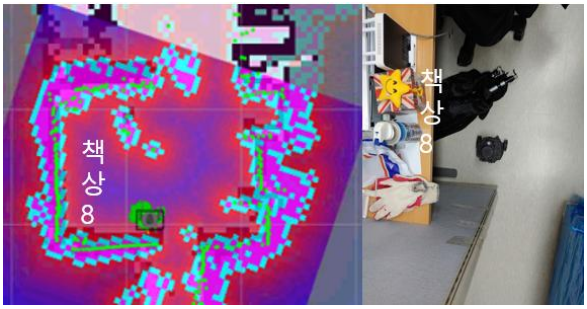

Fig. 8. Desk 8

그림8은 책상8에서의 터틀봇3의 실제 모습과 RVIZ창에서의 터틀봇3의 모습이다. 목표 좌표 x=-3.36, y=0.77이며 도착 좌표 x=-3.93, y=0.79이다. x축으로 0.33, y축으로 0.02오차가 있다. 책상8의 위치는 잘 추정하였지만 반대쪽 벽을 제대로 인식을 못하여서 오차가 발생하였다.

III. Conclutions

불안정한 환경에서 SLAM 방식들 중에서 hector을 이용하여 맵 빌딩을 하였다. 맵 빌딩을 진행 후 네비게이션으로 목표 지점으로 정확히 이동이 가능한지 테스트를 하여 맵 빌딩이 제대로 진행되었는지 테스트를 하였다. 임의의 장애물이나 높이가 다른 형태, 정형화되어 있지 않은 크기의 장애물 등에서 오차를 발생시켰다. 차후 좀더 불확실한 환경에서 그 원인과 대처방안을 연구하고자 한다.

ACKNOWLEDGEMENT

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음(2018-0-00209)

REFERENCES

[1 Turtlebot3 e-Manual ROS1 Navigation Guide , *http://emanual.robotis.com/docs/en/platform/turtlebot3/navigation/#run-navigation-nodes*

[2] Erico Guizzo, “The TurtleBot3 Teacher,” IEE Spectrum, Vol. 54, issue 8, pp19-20, Aug.2017.

[3] Pierre-Yves Lajoie, “DOOR-SLAM: Distributed, Online, and Outlier Resilient SLAM for Robotic Teams,” IEEE Robotics and Automation Letters, Vol. 5, issue 2, pp1656-1663, April.2020.

[4] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ROS-based visual SLAM methods in homogeneous indoor environment," 14th Workshop on Positioning, Navigation and Communications (WPNC), Bremen, 2017, pp. 1-6, doi: 10.1109/WPNC.2017.8250081.

[5] Navigation, *https://github.com/aniskoubaa/gaitech_edu*