

만일 실험을 마치려면, terminal은 CTL+C로 종단 후에 종료

ROS Programming (SLAM)

(file #4 / ?) ver1.0 Noetic

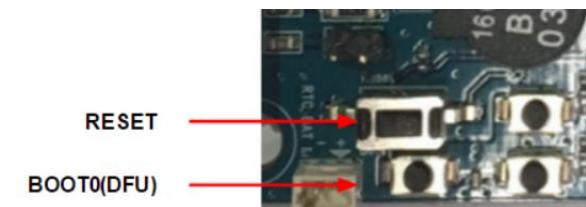
Battery 사전에 충전하기

(미충전 경우 Buzzer 소리 발생)

→ 방치 시 완전 방전됨



Yongseok Chi



만일 회전 안하면, openCR의 RESET누르고, Ctl+c, 새 터미널
`roslaunch turtlebot3_bringup turtlebot3_robot.launch`

TIP

새 터미널 열기 Ctrl + Alt + T

- * 수평 분할 : Ctrl + Shift + O
- * 수직 분할 : Ctrl + Shift + E
- * 다음 창 활성화 : Ctrl + Tab or Ctrl + Shift + N
- * 이전 창 활성화 : Ctrl + Shift + Tab or Ctrl + Shift + P
- * 현재 활성화 된 창 닫기 : Ctrl + Shift + W
- * 터미네이터 실행 : Ctrl + Alt + T
- * 터미네이터 종료 : Ctrl + Shift + Q

The screenshot shows a terminal window with four tabs open, each displaying ROS log output. The tabs are labeled with their respective terminal sessions:

- Tab 1: yongseok@yongseok: ~ 49x17
- Tab 2: pi@raspberrypi: ~ 52x17
- Tab 3: yongseok@yongseok: ~ 52x17
- Tab 4: yongseok@yongseok: ~ 47x17

The content of the tabs includes:

- Tab 1: ROS master startup logs, including process IDs (5093, 5906) and ROS_MASTER_URI.
- Tab 2: ROS master startup logs, including process IDs (5093, 5906) and ROS_MASTER_URI.
- Tab 3: ROS master startup logs, including process IDs (5093, 5906) and ROS_MASTER_URI.
- Tab 4: ROS master startup logs, including process IDs (5093, 5906) and ROS_MASTER_URI.

Turtlebot3 burger : Standard Specification

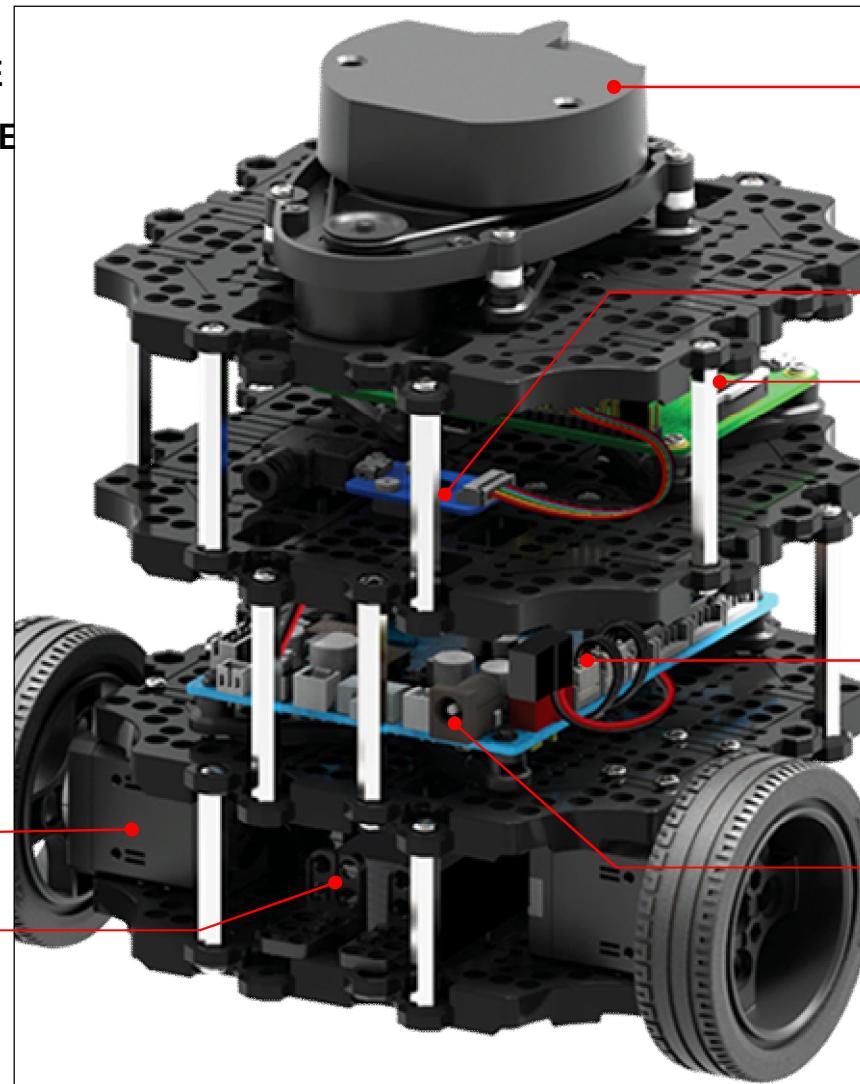
1. Specification

- (1) OPEN SOURCE SOFTWARE
- (2) OPEN SOURCE HARDWARE

DYNAMIXEL

- : XL430-W350-T
- : Stall Torque
 - 4.1 [N.m] (at 12.0 [V], 2.3 [A])
- : No Load Speed
 - 46 [rev/min] (at 12.0 [V])

Li-Po battery 11.1V 1800mAh



360° LiDAR

- : HLS-LFCD LDS(Laser Distance Sensor)
- : Detection distance 120mm~3500mm
- : Angular Resolution 1 degree

: USB2LDS (115200 baudrate)

SBC(Single Board Computer)

- : Raspberry PI 3 B+ / [PI 4](#)
- Broadcom BCM2837B0 / [2711](#)
- Cortex-A53 64bits Quad-core
- 1.4GHz frequency

OpenCR(Control module for ROS)

- : [STM32F746](#)
- ARM Cortex-M7 32bits RISC core
- 216MHz frequency

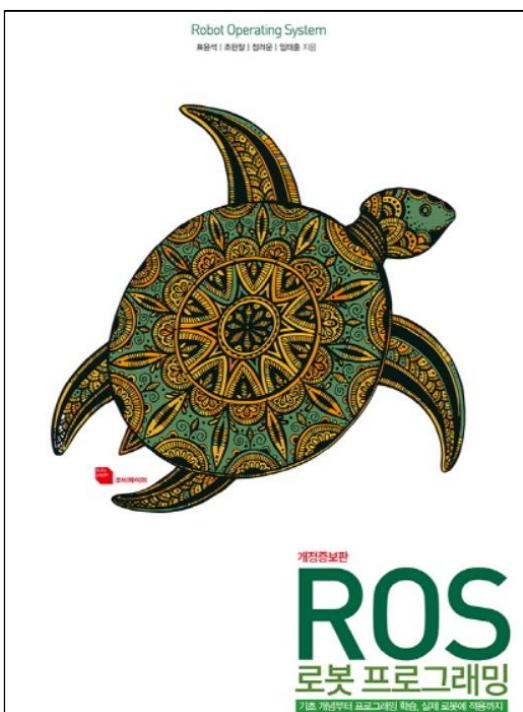
Input 12V 5A

Reference books

: ROS 로봇 프로그래밍 : Ruby paper

: <https://wiki.ros.org>

: Robotis e-manual 4장 <https://emanual.robotis.com/docs/en/platform/turtlebot3/slam/>



A screenshot of the Robotis e-Manual website. The header features the 'ROBOTIS e-Manual' logo and navigation links for DYNAMIXEL, DYNAMIXEL SYSTEM, EDUCATIONAL KITS, SOFTWARE, PARTS, and FAQ. Below the header is a search bar and a menu bar with tabs for Kinetic, Melodic, Noetic, Dashing, Foxy, Humble, and Windows. The main content area is titled '4. SLAM'. It includes a 'NOTE' section with instructions: 'Please run the SLAM on Remote PC.' and 'Make sure to launch the Bringup from TurtleBot3 before executing any operation.' Below the note is a description of SLAM as a technique for drawing maps. To the right of the text is a video thumbnail showing a robot navigating a room with a map overlay, with the caption 'TurtleBot3 09 SLAM using Gmapping and Carto...' and a play button. At the bottom of the page, there's a section titled '4. 1. Run SLAM Node' with a terminal command '\$ roscore' and a note about launching the Bringup if it's not running on the SBC.

6. SLAM (Simultaneous Localization and Mapping)

※ Turtlebot series

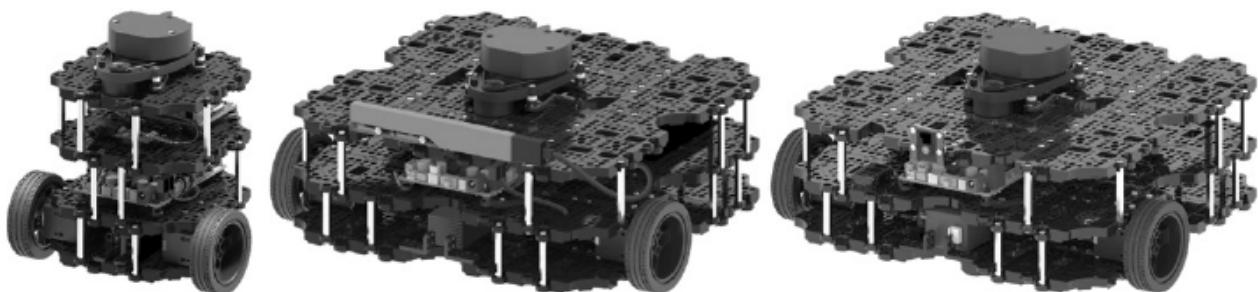
: Turtlebot1 iRobot社

(2010년)



: Turtlebot3 로보티즈社

(2017년)



: Turtlebot2 유진 로봇社

(2012년)

: Turtlebot4 for ROS2

<https://clearpathrobotics.com/turtlebot-4>



6. SLAM (Simultaneous Localization and Mapping)

- SLAM : 동시적 위치 추정 및 지도작성
: to draw a map by estimating current location in an arbitrary space

- LIDAR : 빛을 통한 감지 및 거리 측정(Light Detection and Ranging)

LIDAR (turtlebot3 – LDS-01) :

https://hlds.co.jp/product/details_en

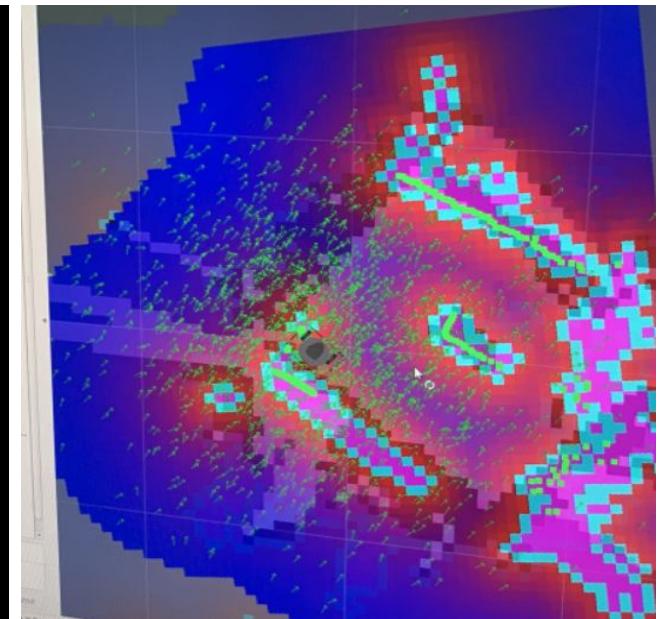
<https://hitachi-lg.com/kr/products/sensor/>

LDS-01

: HLS-LFCD-LDS(Laser Distance Sensor)

: HLDS(Hitachi-LG Data Storage) 개발

: 2D laser scanner capable
of sensing 360 degrees



6. SLAM (Simultaneous Localization and Mapping)

- LIDAR (turtlebot3) : <https://hitachi-lg.com/kr/products/sensor/>

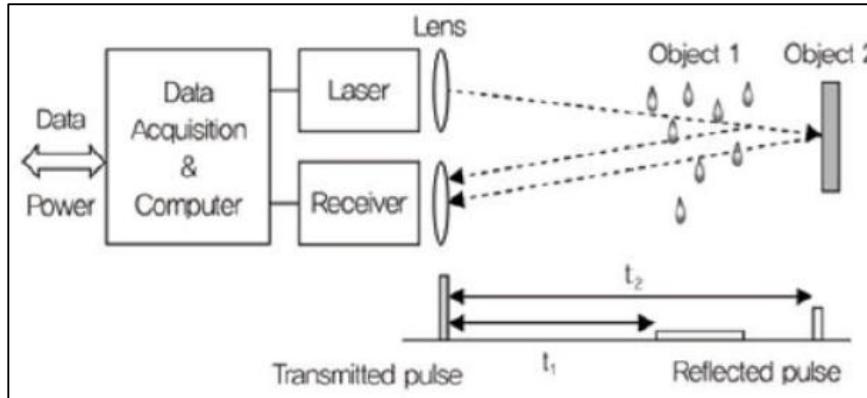
항목	LDS 1.5	항목	LDS 2.0
동작 공급 전압	5VDC±5%	동작 공급 전압	5VDC±5%
광원	Semiconductor Laser Diode($\lambda=785\text{nm}$)	광원	Semiconductor Laser Diode($\lambda=785\text{nm}$)
레이저 안전	IEC60825-1 Class 1	레이저 안전	IEC60825-1 Class 1
소비 전류	400mA 이하 (돌입 전류 1A)	소비 전류	500mA 이하 (돌입 전류 1A)
탐지 거리	120mm ~ 3500mm	탐지 거리	120mm ~ 5000mm
거리 정확도	0.12m ~ 0.49m : ±15mm 0.50m ~ 3.50m : ±5.0%	거리 정확도	0.12m ~ 0.49m : ±15mm 0.50m ~ 5.00m : ±5.0%
거리 정밀도	0.12m ~ 0.49m : 10mm 미만 0.50m ~ 3.50m : 3.5% 미만	거리 정밀도	0.12m ~ 0.49m : 10mm 미만 0.50m ~ 5.00m : 3.5% 미만
주사 속도	300±10 rpm	주사 속도	300±10 rpm
각도 범위	360°	각도 범위	360°
각도 분해능	1°	각도 분해능	1°
인터페이스	3.3V USART (230,400 bps) <u>6도 당 42 바이트, 전 이중 방식 옵션</u>	인터페이스	3.3V USART (115,200 bps) <u>4도 당 22 바이트, 전 이중 방식 옵션</u>
주위 내광성	10,000 lux 이하	주위 내광성	10,000 lux 이하
Sampling Rate	1.8kHz	샘플링 레이트	1.8kHz
크기	69.5(W) X 95.5(D) X 39.5(H)mm	크기	79.7(W) X 108(D) X 54.5(H)mm
질량	125g 미만	질량	150g 미만
소음	특별한 소음이 없음	소음	특별한 소음이 없음



6. SLAM (Simultaneous Localization and Mapping)

● ToF(Time of Flight) 방식

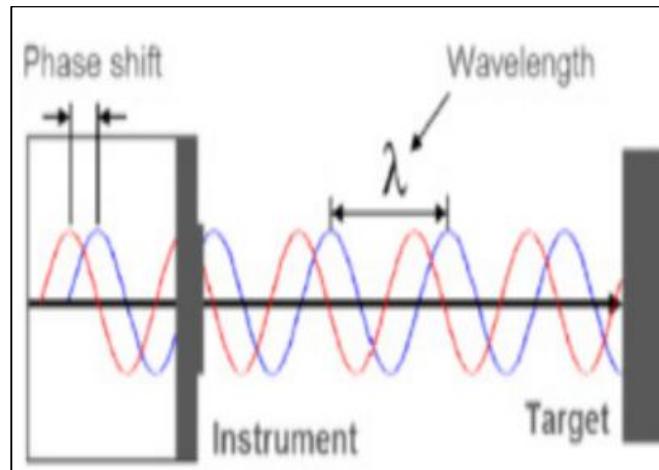
: ToF 방식은 Laser 펄스 신호가 측정 범위 내의 물체에서 반사되어 수신기에 도착한 시간을 측정하여 거리를 측정하는 원리



Ref. <https://www.hellot.net/mobile/article.html?no=43483>

● PS(Phase Shift) 방식

: 특정 주파수를 가지고 연속적으로 변조되는 Laser 빔을 방출하고, 물체에서 반사되어 오는 Laser 신호의 위상 변화량을 측정하여 거리를 측정하는 방식



Ref. Time-of-Flight 방식과 Phase Shift 방식 LiDAR의 현장 적용성 기초연구

6. SLAM (Simultaneous Localization and Mapping)

6-1. Specification of LiDAR

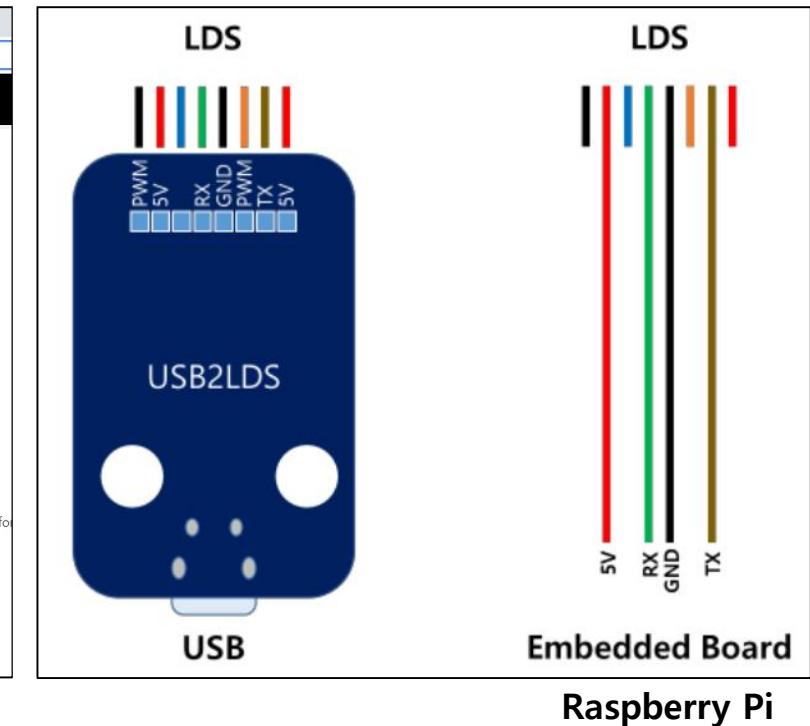
(1) 360° LiDAR : HLS-LFCD LDS(Laser Distance Sensor)

- Detection distance **120mm~3500mm**
- Angular Resolution **1 degree** / USB2LDS (**UART 통신 115,200 baudrate**)
- Robotis e-manual 13장 https://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lidar_01/

The screenshot shows a web browser displaying the Robotis e-Manual. The URL is https://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lidar_01/. The page title is "13. 1. 3. LDS-01" and the sub-section is "13. 1. 3. 1. Overview". On the left, there is a sidebar with a navigation menu for TurtleBot3, including sections like Overview, Features, Quick Start Guide, SLAM, Navigation, Simulation, Manipulation, Autonomous Driving, Machine Learning, Examples, Friends(Locomotion), Learn, More Info, and FAQ. The main content area features two images of the LDS-01 sensor: a top-down view and a side view. Below the images is a bulleted list of specifications:

- 360 Laser Distance Sensor LDS-01 is a 2D laser scanner capable of sensing 360 degrees that collects a set of data around the robot to use for localization and Navigation.
- The LDS-01 is used for TurtleBot3 Burger, Waffle and Waffle Pi models.
- It supports USB interface(USB2LDS) and is easy to install on a PC.
- It supports UART interface for embedded board.

At the bottom of the page, there is a link to "13. 1. 3. 2. Introduction Video" and a note "[Video #01] How to use the LDS-01".

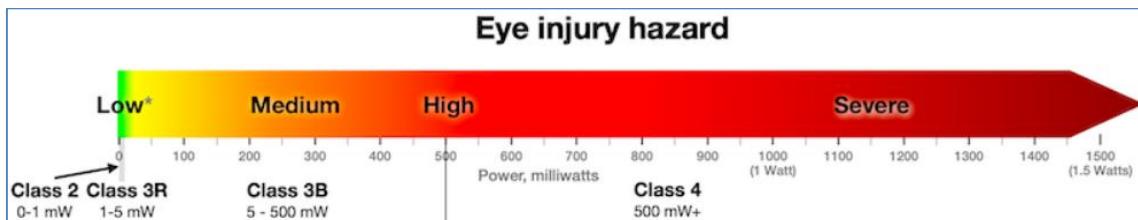


6. SLAM (Simultaneous Localization and Mapping)

(2) 360° LiDAR : HLS-LFCD LDS(Laser Distance Sensor)

Items	Specifications	Items	Specifications
Operating supply voltage	5V DC ±5%	Distance Range	120 ~ 3,500mm
Light source	Semiconductor Laser Diode($\lambda=785\text{nm}$)	Distance Accuracy (120mm ~ 499mm)	±15mm
LASER safety	IEC60825-1 Class 1	Distance Accuracy(500mm ~ 3,500mm)	±5.0%
Current consumption	400mA or less (Rush current 1A)	Distance Precision(120mm ~ 499mm)	±10mm
Detection distance	120mm ~ 3,500mm	Distance Precision(500mm ~ 3,500mm)	±3.5%
Interface	3.3V USART (230,400 bps) 42bytes per 6 degrees, Full Duplex option	Scan Rate	300±10 rpm
Ambient Light Resistance	10,000 lux or less	Angular Range	360°
Sampling Rate	1.8kHz	Angular Resolution	1°
Dimensions	69.5(W) X 95.5(D) X 39.5(H)mm		

Laser safety classes : <https://www.lasersafetyfacts.com/laserclasses.html>



6. SLAM (Simultaneous Localization and Mapping)

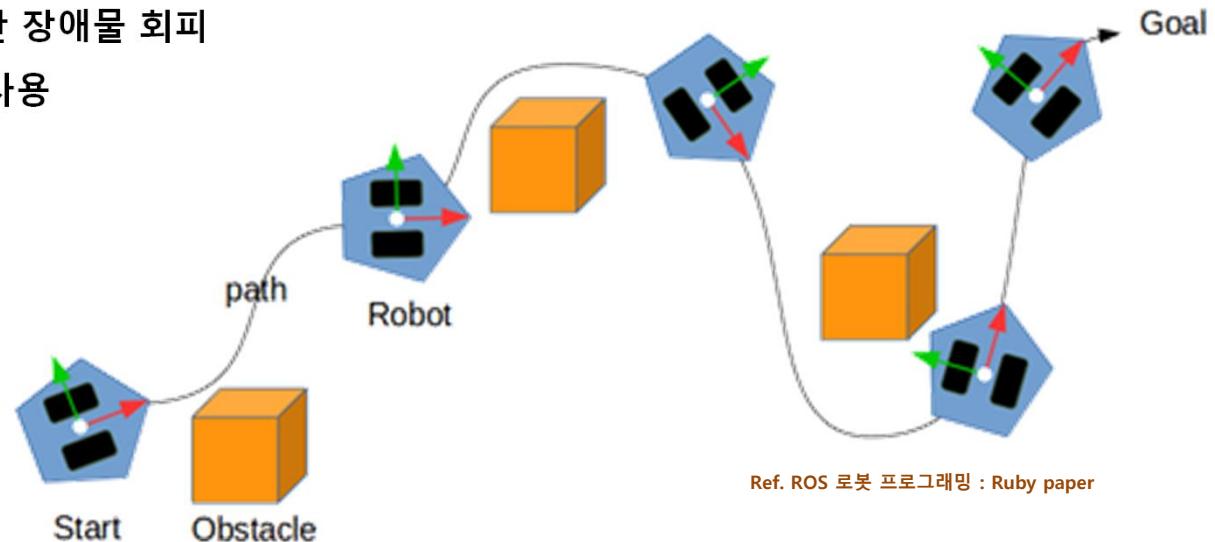
6-2. SLAM Node

※ Navigation

- ① 위치: 로봇의 위치 계측/추정하는 기능 → **Localization, pose estimation** 라 부름
- ② 센싱: 벽, 물체 등의 장애물의 계측하는 기능 → **sensing**
- ③ 지도: 길과 장애물 정보가 담긴 지도
- ④ 경로: 목적지까지 최적 경로를 계산하고 주행하는 기능
→ **motion planning(path planning)**라고 부름
- ⑤ 장애물 회피(collision avoidance)
 - : 이동 궤적에 따라 목적지까지 이동중 갑자기 나타난 장애물 회피
 - **Dynamic Window Approach(DWA)** 알고리즘 사용

※ 위치 검출

- 실외 GPS (Global Positioning System)
 - 실내 Indoor Positioning Sensor
 - : Landmark (Color, IR Camera)
 - : Indoor GPS, WiFi SLAM, Beacon
- 실내 위치 검출의 경우 오차 발생



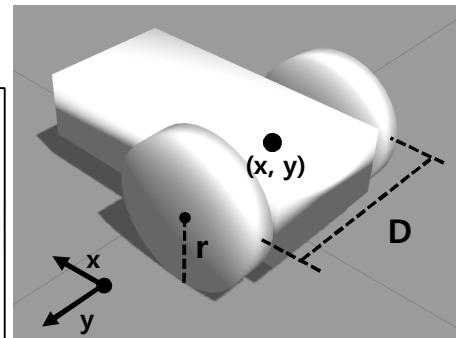
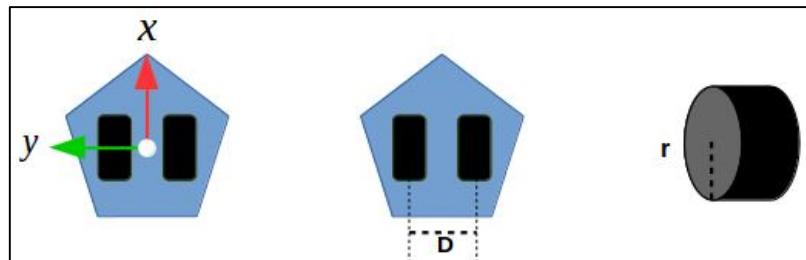
6. SLAM (Simultaneous Localization and Mapping)

6-2. SLAM Node

※ 로봇의 위치 계측/추정하는 기능

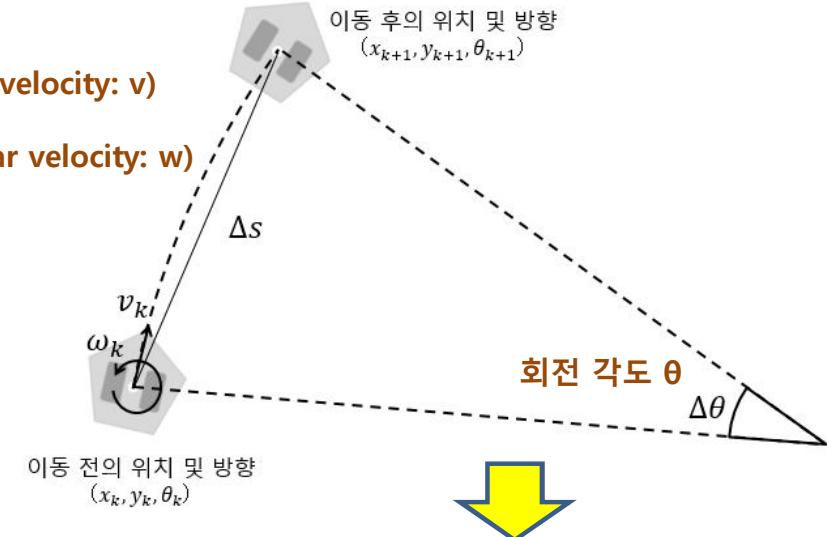
① 추측 항법(dead reckoning)

- 양 바퀴 축의 회전 값을 이용 (주행기록계 odometer)
- 이동 거리와 회전 값을 계산, 위치 측정
- 바닥 슬립, 기계적, 누적 오차 발생
- IMU(자이로/가속도/지자계) 등 관성 센서 및 필터로 위치 보상
- Kalman filter
- 필요한 정보(encoder: wheel의 회전량 측정)
 - : 양 바퀴 축의 엔코더 값 E
 - : 바퀴 간 거리 D, 바퀴 반지름 r



Ref. ROS 로봇 프로그래밍 : Ruby paper

선속도
(linear velocity: v)
각속도
(angular velocity: w)



로봇 이동 후의
위치 $x_{(k+1)}, y_{(k+1)}$ 및 방향 $\Theta_{(k+1)}$

$$x_{(k+1)} = x_k + \Delta s \cos \left(\theta_k + \frac{\Delta\theta}{2} \right)$$
$$y_{(k+1)} = y_k + \Delta s \sin \left(\theta_k + \frac{\Delta\theta}{2} \right)$$
$$\theta_{(k+1)} = \theta_k + \Delta\theta$$

6. SLAM (Simultaneous Localization and Mapping)



turtlebot_core

: calcOdometry 함수

```
bool calcOdometry(double diff_time)
{
    float* orientation;
    double wheel_l, wheel_r;
    double delta_s, theta, delta_theta;
    static double last_theta = 0.0;
    double v, w;
    double step_time;

    wheel_l = wheel_r = 0.0;
    delta_s = delta_theta = theta = 0.0;
    v = w = 0.0;
    step_time = 0.0;

    step_time = diff_time;

    if (step_time == 0)
        return false;

    wheel_l = TICK2RAD * (double)last_diff_tick[LEFT];
    wheel_r = TICK2RAD * (double)last_diff_tick[RIGHT];

    if (isnan(wheel_l))
        wheel_l = 0.0;

    if (isnan(wheel_r))
        wheel_r = 0.0;

    delta_s    = WHEEL_RADIUS * (wheel_r + wheel_l) / 2.0;
    orientation = sensors.getOrientation();
    theta      = atan2f(orientation[1]*orientation[2] +
                        orientation[0]*orientation[3],
                        0.5f - orientation[2]*orientation[2] -
                        orientation[3]*orientation[3]);

    delta_theta = theta - last_theta;

    // compute odometric pose
    odom_pose[0] += delta_s * cos(odom_pose[2] + (delta_theta / 2.0));
    odom_pose[1] += delta_s * sin(odom_pose[2] + (delta_theta / 2.0));
    odom_pose[2] += delta_theta;

    // compute odometric instantaneouse velocity
    v = delta_s / step_time;
    w = delta_theta / step_time;

    odom_vel[0] = v;
    odom_vel[1] = 0.0;
    odom_vel[2] = w;

    last_velocity[LEFT] = wheel_l / step_time;
    last_velocity[RIGHT] = wheel_r / step_time;
    last_theta = theta;

    return true;
}
```

6. SLAM (Simultaneous Localization and Mapping)

6-2. SLAM Node

※ 센싱: 벽, 물체 등의 장애물의 계측하는 기능

- 거리센서, 비전센서, Depth 카메라

※ 지도: 길과 장애물 정보가 담긴 지도

- SLAM(Simultaneous Localization And Mapping)

※ 경로: 목적지까지 최적 경로를 계산하고 주행하는 기능 ★

- 내비게이션(Navigation)
- 위치 추정 (Localization / Pose estimation)
- 경로 탐색/계획 (Path search and planning)

: Dynamic Window Approach (DWA)

→ 이동 경로 계획 및 장애물을 회피할 때 사용되는 방법(성능 탁월)

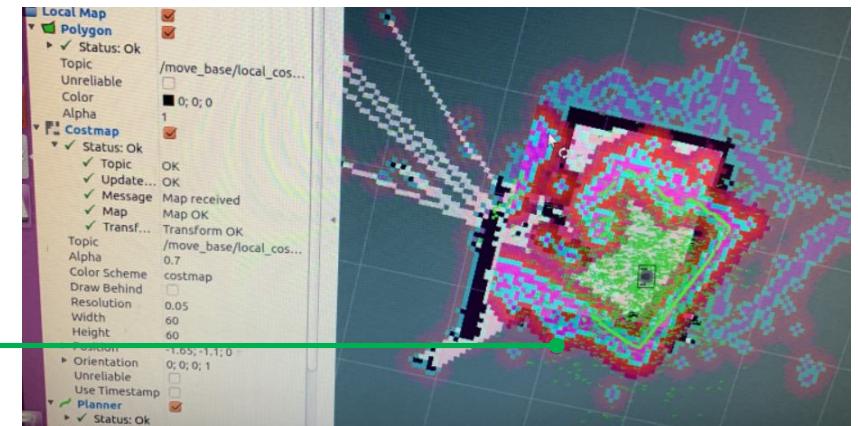
→ 로봇의 속도 탐색 영역(velocity search space)에서, 로봇과 충돌 가능한 장애물을 회피하면서

목표점까지 빠르게 다다를 수 있는 속도를 선택하는 방법

: A* (A Star) 알고리즘

: Potential Field

: Particle Filter ★



6. SLAM (Simultaneous Localization and Mapping)

6-2. SLAM Node

(1) SLAM(Simultaneous Localization and Mapping)

: 동시적 위치 추정 및 지도작성

: to draw a map
by estimating current location
in an arbitrary space

: OpenSLAM

- <https://openslam-org.github.io/>
- SLAM package 종류

→ gmapping (터틀봇 적용)

(<http://wiki.ros.org/gmapping>)

Melodic version 지원 안됨

→ cartographer (ROS2)

→ hector mapping

→ rtabmap

The screenshot shows a web browser window with the URL openslam-org.github.io. The page displays a sidebar on the left listing various SLAM projects and tools, with a red box highlighting the 'GMapping' entry. The main content area shows the 'GMapping' project page, which includes sections for Long Description, Example Images, Input Data, Logfile Format, Type of Map, Hardware/Software Requirements, and Papers Describing the Approach. The 'OpenSLAM' logo is visible at the top right.

AM.org?

AM) problem has been intensively different techniques have been s implementations to the provide a platform for SLAM publish their algorithms. 2018, it has been moved to github.

Enter Search Terms



Kinetic Melodic Noetic Dashing Foxy Humble Windows

TurtleBot3

- 1. Overview >
 - 2. Features >
 - 3. Quick Start Guide >
 - 4. SLAM <ul style="list-style-type: none;"> - 4. 1. Run SLAM Node
 - 4. 2. Run Teleoperation Node
 - 4. 3. Tuning Guide
 - 4. 4. Save Map
 - 4. 5. Map
5. Navigation >
6. Simulation >
7. Manipulation >
8. Autonomous Driving >
9. Machine Learning >
10. Examples >
11. Friends(Locomotion) >
12. Learn >
13. More Info >
14. FAQ

📌 How to save the TURTLEBOT3_MODEL parameter?

📌 Read more about other SLAM methods

- **Gmapping** ([ROS WIKI](#), [Github](#))

1. Install dependent packages on PC.
Packages related to Gmapping have already been installed on [PC Setup](#) section.
2. Launch the Gmapping SLAM node.

```
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
```

- **Cartographer** ([ROS WIKI](#), [Github](#))

1. Download and build packages on PC.

The Cartographer currently does not provide the binary installation method for ROS1 Noetic. Please download and build refer to [official wiki page](#) for more details.

```
$ sudo apt update  
$ sudo apt install -y python3-wstool python3-rosdep ninja-build stow  
$ cd ~/catkin_ws/src  
$ wstool init src  
$ wstool merge -t src https://raw.githubusercontent.com/cartographer-project/cartographer_ros/master/cartographer_ros.rosinstall  
$ wstool update -t src  
$ sudo rosdep init  
$ rosdep update  
$ rosdep install --from-paths src --ignore-src --rosdistro=noetic -y  
$ src/cartographer/scripts/install_abseil.sh  
$ sudo apt remove ros-noetic-abseil-cpp  
$ catkin_make_isolated --install --use-ninja
```

2. Launch the Cartographer SLAM node.

```
$ source ~/catkin_ws/install_isolated/setup.bash  
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=cartographer
```

- **Karto** ([ROS WIKI](#), [Github](#))

1. Install dependent packages on PC.

```
$ sudo apt install ros-noetic-slam-karto
```

2. Launch the Kart SLAM node.

```
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=karto
```

6. SLAM (Simultaneous Localization and Mapping)

6-2. SLAM Node

(2) Gmapping

- : OpenSLAM에 공개(free)된 SLAM 종류, ROS package로 제공
- : 최근 가장 많이 사용되는 물체 추정 알고리즘으로 Rao-Blackwellized 파티클 필터(Particle filter) 사용
 - Particle 수가 감소되었고, grid 맵 제공
 - X, Y(x,y) 병진속도 linear velocity), Theta 속도(angular velocity) 이동 명령을 받도록 되어있어
 - 두 축의 구동 모터로 이루어져 좌/우측 바퀴를 따로 구동하는 로봇이나 3개 이상의 구동 휠을 가지고 있어야 함
 - 주행기록계(Odometry)가 반드시 있어 이동한 거리, 회전량을 추측 항법(dead reckoning)으로 계산할 수 있어야 함
 - 자세(pose) 보상, 위치 계측 및 추정 수단이 있어야 됨
 - XY 평면상의 장애물을 계측하기 위해
 - LDS(Laser Distance Sensor), LRF(Laser Range Finder), LiDAR(Light Detection And Ranging) 있어야 됨
 - 로봇 형태 제약 조건
 - 정다각형, 직사각형, 원형 로봇 적합
 - 한쪽 축이 길게 나온 변형 로봇, 문 사이로 나가지 못할 정도로 큰 로봇, 휴머노이드, 다관절 이동 로봇, 비행 로봇 제외
(휴머노이드, 다관절의 경우 자세 계측이 어려움)

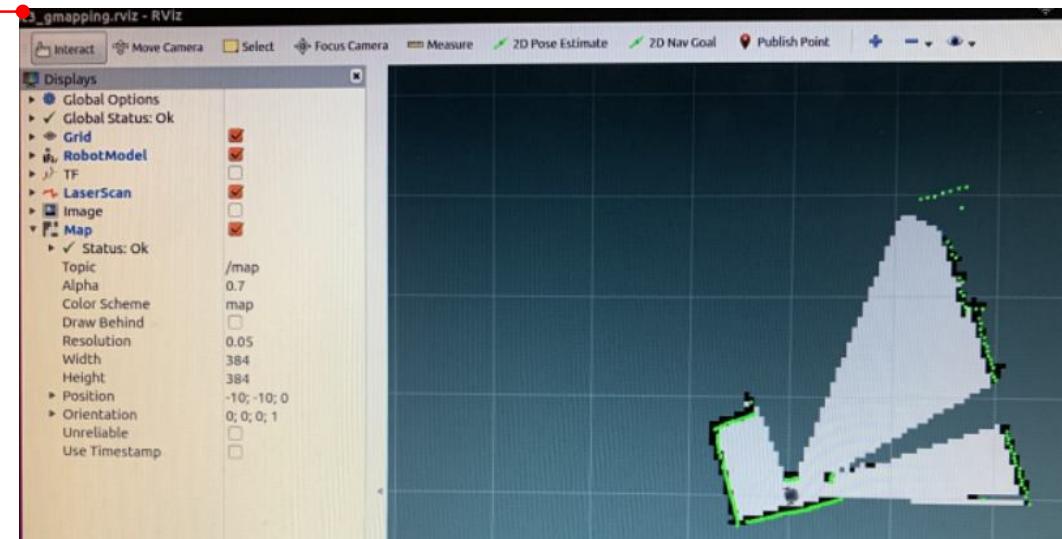
6. SLAM (Simultaneous Localization and Mapping)

6-2. SLAM Node

(2) Gmapping

: Gmapping 계측 환경

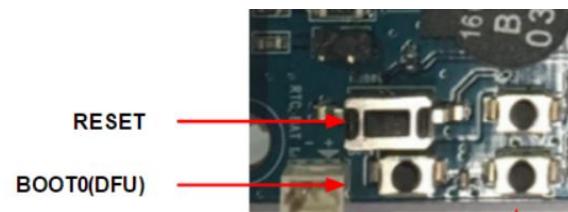
- 다음의 환경에 적합하지 않음
 - 장애물 하나 없는 정사각형 형태의 환경
 - 장애물 없는 평행한 복도
 - 레이저 및 적외선이 반사되지 못하는 유리창
 - 레이저 및 적외선이 산란하는 거울
 - 호수, 바닷가 기타 물가



(3) Cartographer

- : Google 제작한 SLAM 기법으로, 2D 및 3D 매핑을 지원
- : 그리드 방식의 매핑과 Ceres 기반 Scan Matcher를 이용하여 환경 재구성
- : Cartographer는 ROS2에서 동작, ROS2를 지원하는 package
- : Gmapping은 라이선스로 인하여 상업용으로 사용하기 어려움
- : Google Cartographer는 Apache 2.0 라이센스를 사용, 지속적인 유지 보수가 되고 있음(현재 가장 많이 사용됨)

6. SLAM (Simultaneous Localization and Mapping)



(4) SLAM 실습

: gmapping SLAM 실습 <http://wiki.ros.org/gmapping>

만일 회전 안하면, openCR의 RESET 누르고, Ctrl+c, 새 터미널
roslaunch turtlebot3_bringup turtlebot3_robot.launch

: 4-1장 <https://emanual.robotis.com/docs/en/platform/turtlebot3/slam/#run-slam-node>

→ R 새 터미널 열고 \$ roscore

R-S 새 터미널 열고 \$ ssh ubuntu@192.168.0.21 (IP is Raspberry, password turtlebot)

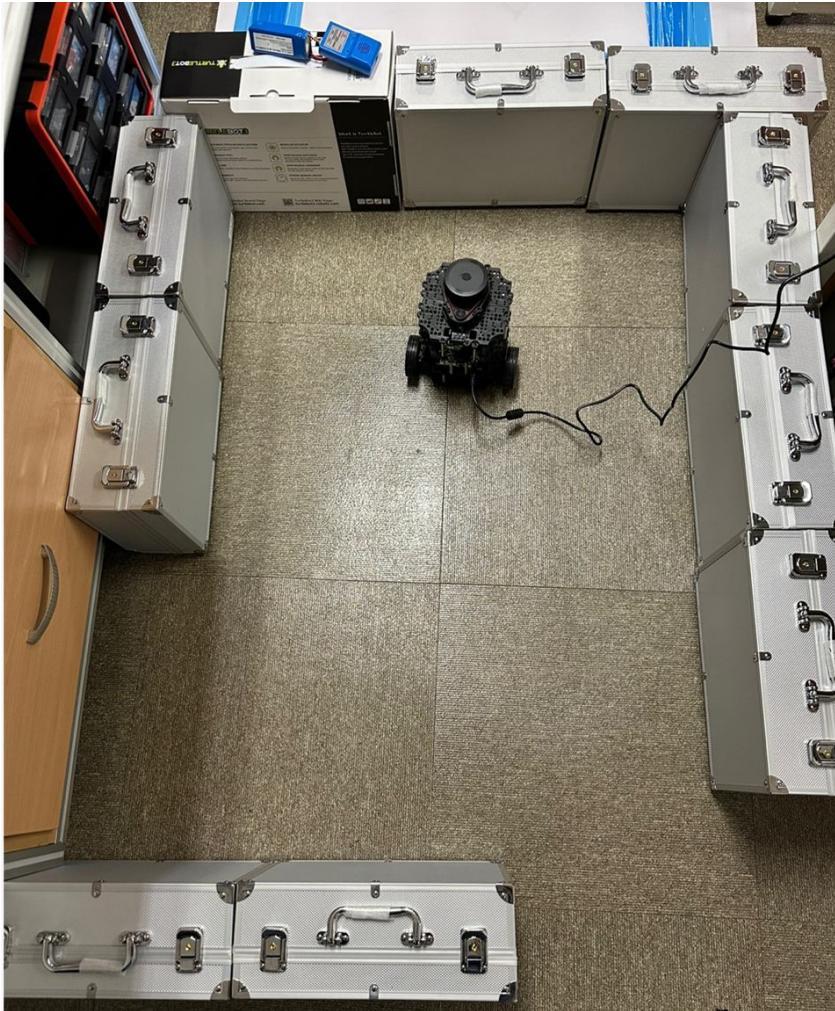
\$ roslaunch turtlebot3_bringup turtlebot3_robot.launch → LDS 회전 확인(회전 안하면)

R 새 터미널 열고
\$ roslaunch turtlebot3_slam turtlebot3_slam.launch

→ SLAM 실행되면서 Rviz 화면이 열리면서 turtlebot과 장애물 화면 생성됨

[실험화면]

Ⓐ turtlebot3를 장애물이 있는 바닥에 내려놓음



Ⓑ 새터미널 열고

① roscore

```
roscore http://192.168.1.141:11311/53x25
yongseok@yongseok: ~ roscore
... logging to /home/yongseok/.ros/log/192.168.1.141-11311-11ee-b171-374fa81075.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt.
Done checking log file disk usage. Usage is <1GB.

started roscore server http://192.168.1.141:39739/
ros_comm version 1.16.0
```

② ssh ubuntu@192.168.0.21
roslaunch turtlebot3_bringup turtlebot3_robot.launch

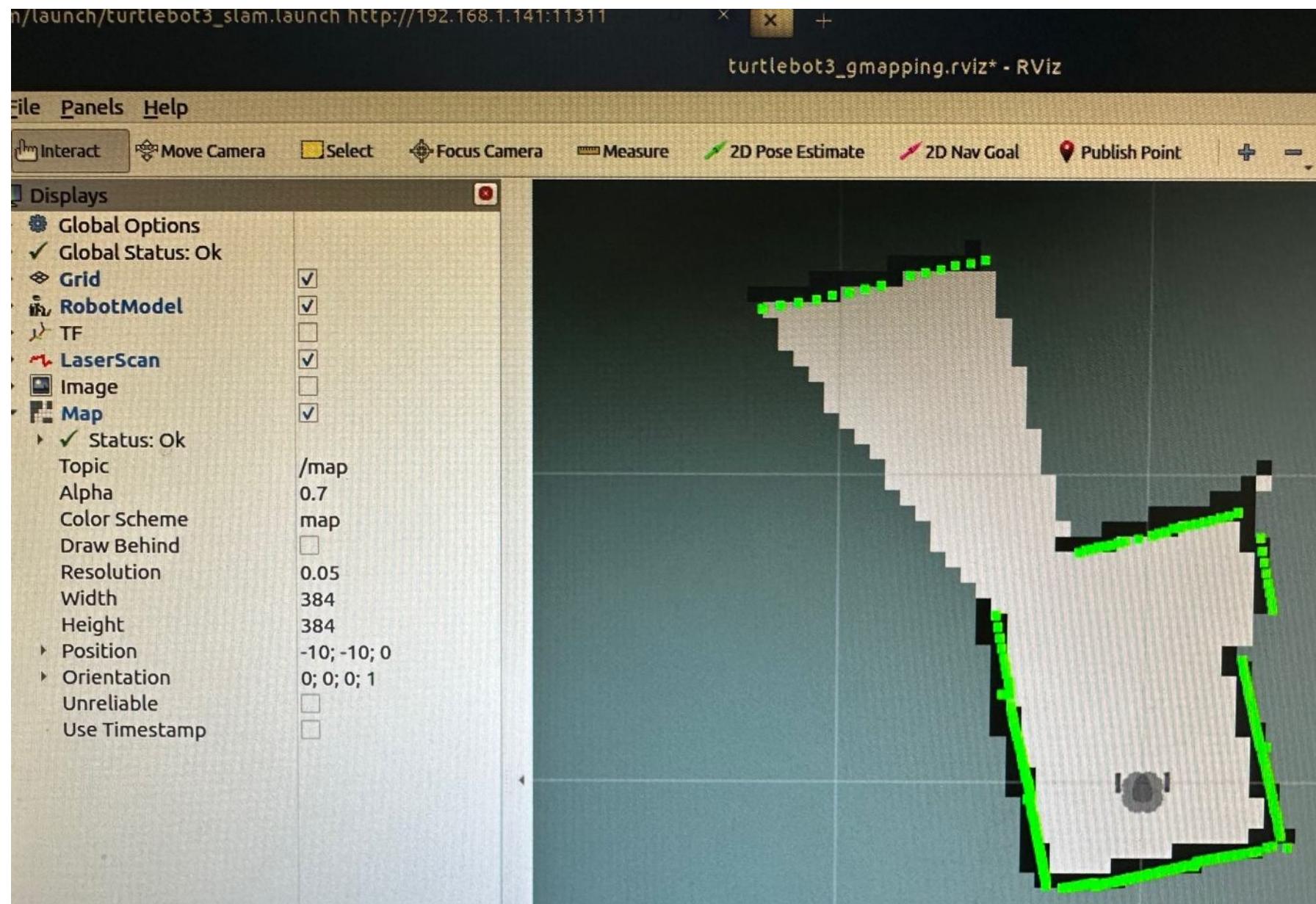
```
[INFO] [1701414455.508898]: Connected to OpenCR board
[INFO] [1701414455.576883]: Connected to OpenCR board
[INFO] [1701414455.587611]: This core(v1.2.6) is compatible with TB3 Burger
[INFO] [1701414455.606383]: ...
```

③ roslaunch turtlebot3_slam turtlebot3_slam.launch

④ Rviz 화면 열림

[실험화면]

④ RViz 화면



6. SLAM (Simultaneous Localization and Mapping)

▶ [Remote PC에서] \$ rosrun turtlebot3_teleop turtlebot3_teleop_key.launch

또는 \$ rosrun turtlebot3_teleop turtlebot3_teleop_key.launch --screen

• rosrun 여러 개의 node가 동시에 실행되기에, node에서 출력되는 message가 보이지 않는 경우 --screen 사용

```
yongseok@yongseok:~$ rosrun turtlebot3_teleop turtlebot3_teleop_key.launch
... logging to /home/yongseok/.ros/log/95a34550-2b1d-11ec-940a-f40669f0af37/ro
.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
```

```
started rosrun server http://192.168.0.20:37125/
```

```
SUMMARY
=====
```

```
PARAMETERS
```

```
* /model: buger
* /rosdistro: kinetic
* /rosversion: 1.12.17
```

```
NODES
```

```
/    turtlebot3_teleop_keyboard (turtlebot3_teleop/turtlebot3_teleop_key)
```

```
ROS_MASTER_URI=http://192.168.0.20:11311
```

```
process[turtlebot3_teleop_keyboard-1]: started with pid [5467]
```

```
Control Your TurtleBot3
```

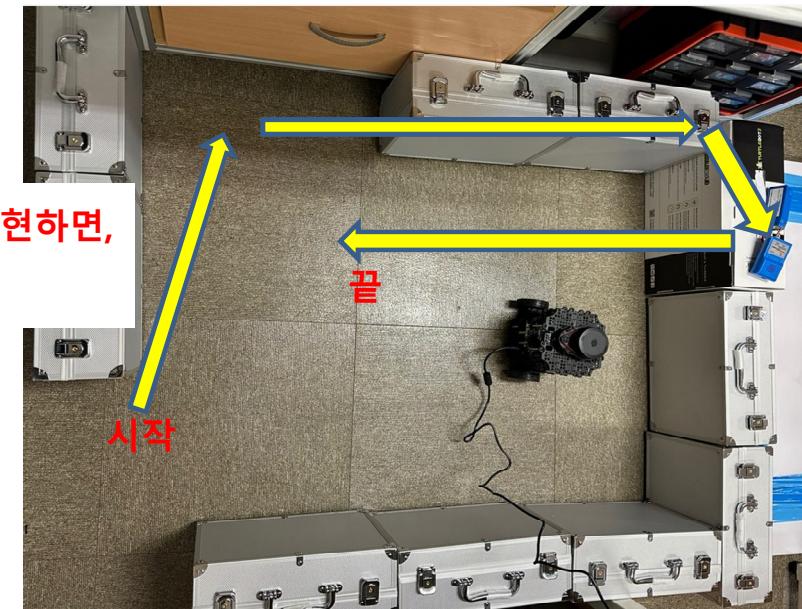
```
Moving around:
```

w		
a	s	d
x		

KEY를 이용하여 아래 외곽을 한바퀴 돌면서 Map을 구현하면,
Rviz에서도 turtlebot이 움직이면서 map 구현됨

w 전진, x 후진, a 왼쪽 회전, d 오른쪽 회전, s 멈출
위 키를 계속 누르면 빨라짐 – 매우 주의

▶ KEY를 이용하여 아래 외곽을 한바퀴 돌면서
Map을 형성한다



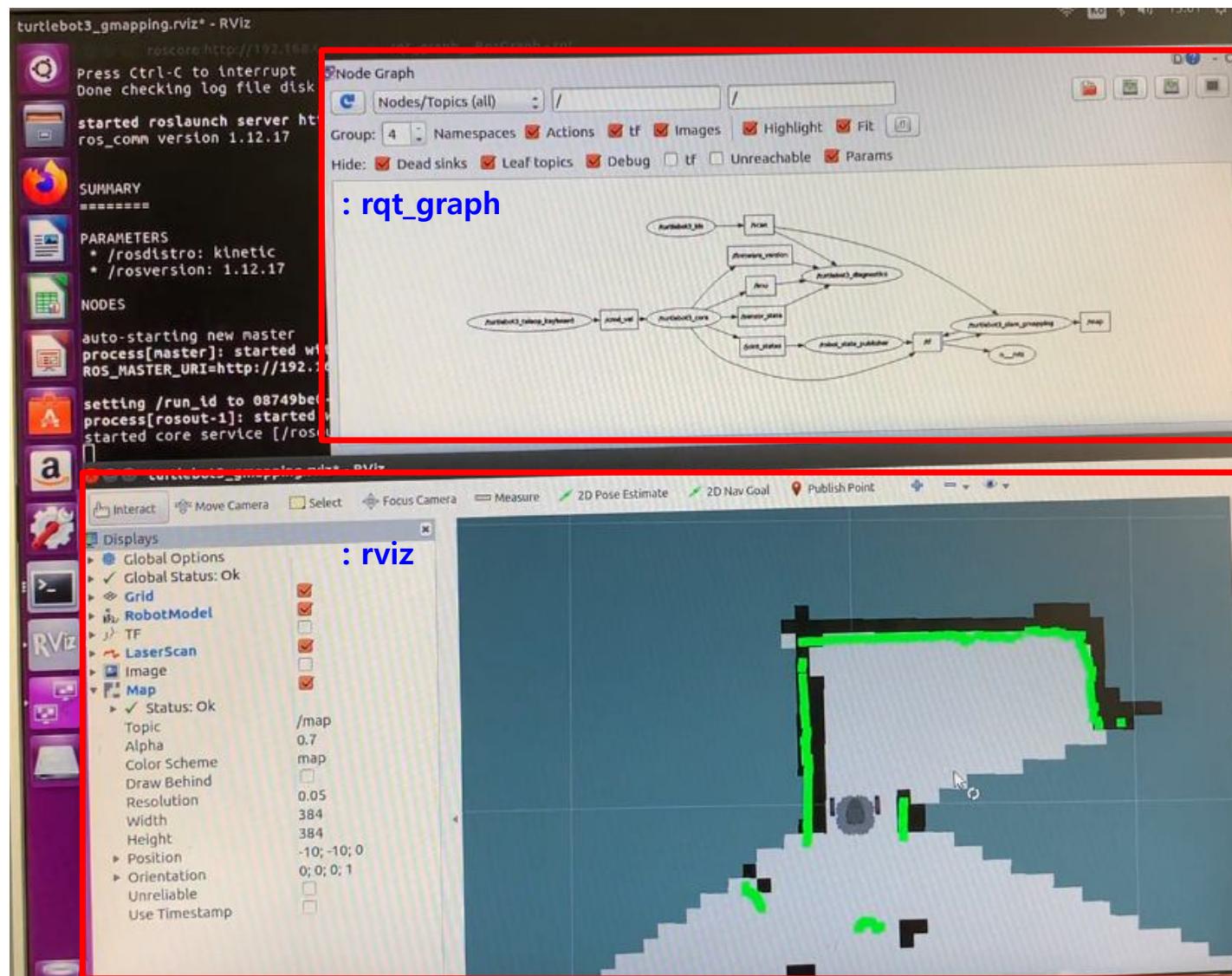
새터미널에서

\$ rqt_graph

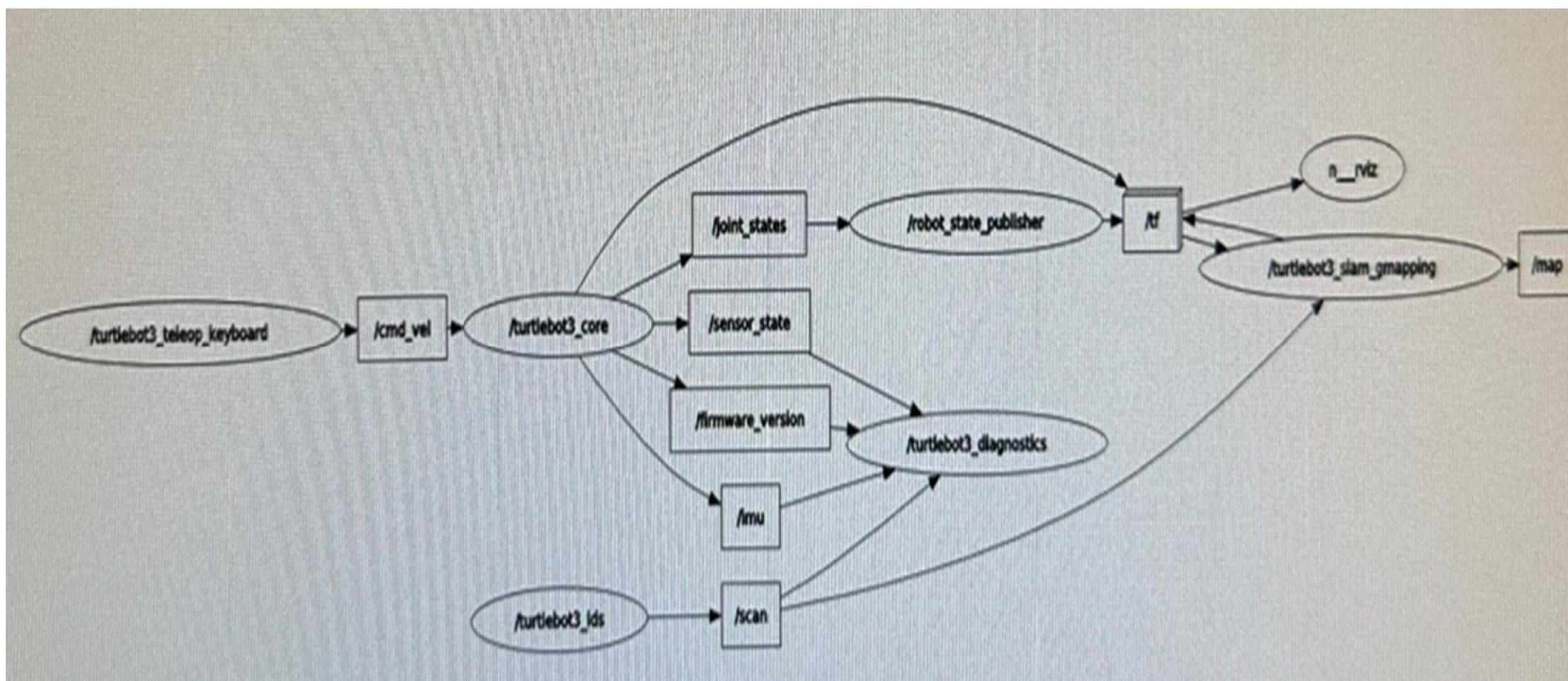
- 실행하여 node간 통신data 확인

6. SLAM (Simultaneous Localization and Mapping)

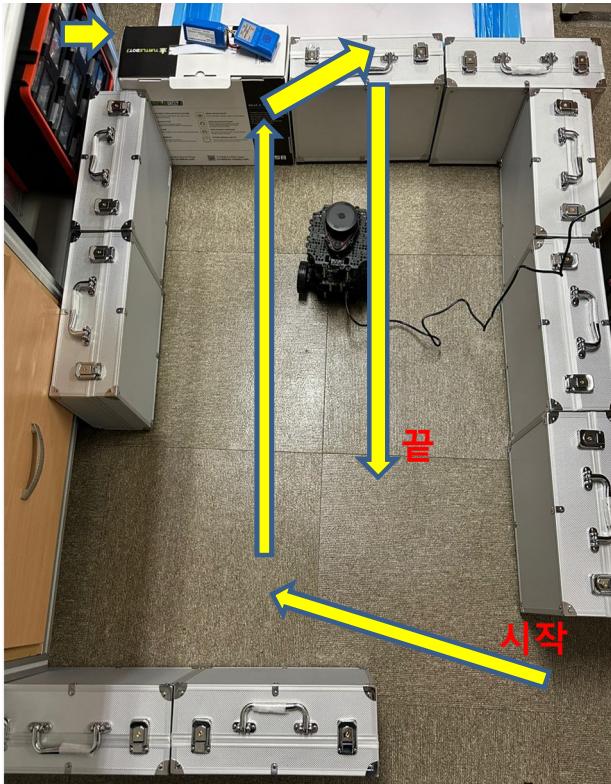
: rqt_graph, rviz 실행 초기 화면



6. SLAM (Simultaneous Localization and Mapping)



6. SLAM (Simultaneous Localization and Mapping)



KEY를 이용하여 아래 외곽을 한바퀴 돌면서 Map을 구현하면,
Rviz에서도 turtlebot이 움직이면서 map 구현됨



6. SLAM (Simultaneous Localization and Mapping)

→ : 맵 구현이 완료되면 아래와 같이 저장함.

Remote PC 에서 새 터미널

\$ rosrun map_server map_saver -f ~/map

-f : option. Map 파일이 저장된 폴더와 이름을 나타냄

-f ~/map : /home/username 폴더에 map으로 저장됨

```
yongseok@yongseok:~$ rosrun map_server map_saver -f ~/map
[ INFO] [1634981609.170774369]: Waiting for the map
[ INFO] [1634981609.440261120]: Received a 384 X 384 map @ 0.050 m/pix
[ INFO] [1634981609.440372279]: Writing map occupancy data to /home/yongseok/map
.pgm
[ INFO] [1634981609.466263768]: Writing map occupancy data to /home/yongseok/map
.yaml
[ INFO] [1634981609.468380584]: Done
yongseok@yongseok:~$
```

완료 확인하기

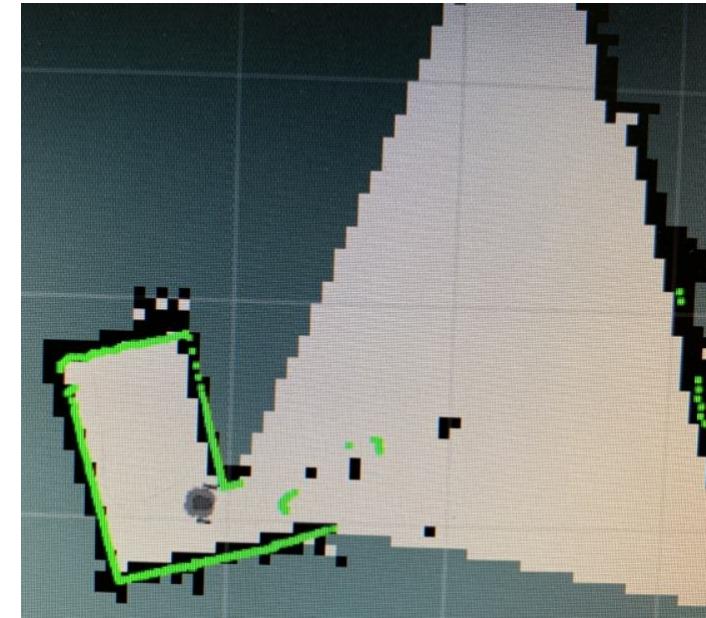
map(map.pgm 파일, map, yaml 파일) 저장 위치 확인

```
yongseok@yongseok:~$ ls
catkin_ws  examples.desktop      Music          Templates
Desktop    install_ros_kinetic.sh parameters.yaml  test
Documents  map.pgm               Pictures       test_linux
Downloads  map.yaml             Public        Videos
yongseok@yongseok:~$
```

6. SLAM (Simultaneous Localization and Mapping)

: gmapping map

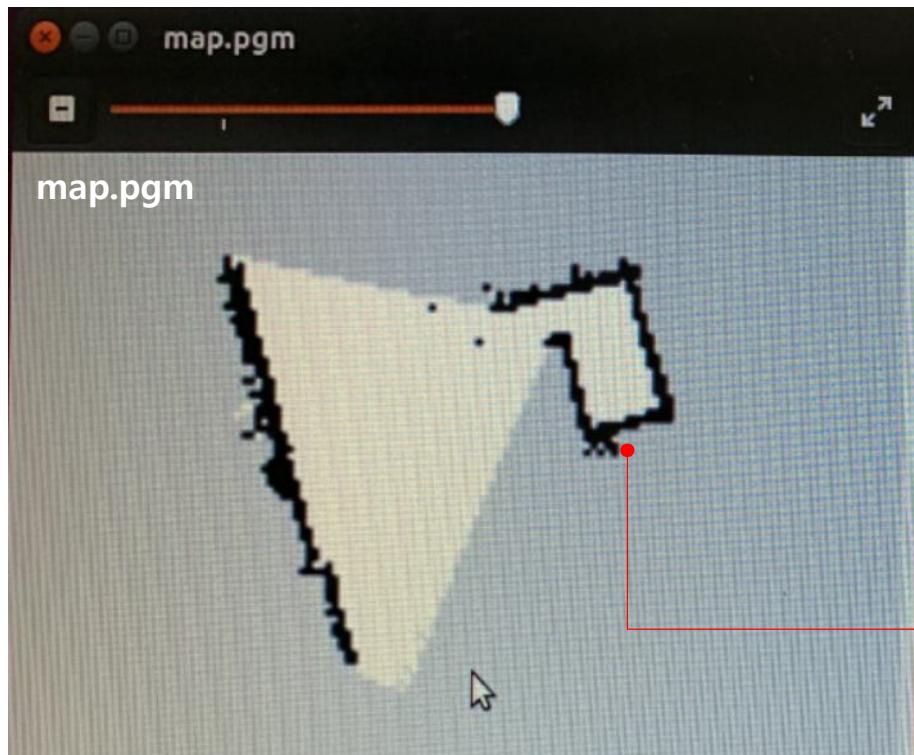
- 2차원 점유 격자 지도(OGM, Occupancy Grid Map)
 - 흰색은 로봇이 이동 가능한 자유 영역(free area)
 - 검은색은 이동 불가능한 점유 영역(occupied area)
 - 회색은 확인되지 않은 미지 영역(unknown area)
- 영역들은 0에서 255의 값으로 표현하는 그레이스케일(gray scale) 값으로 나타냄
- 0~255는 점유 상태(occupancy state)를 표현한 점유 확률(occupancy probability)
 - 베이즈(Bayes)정리의 사후 확률(posterior probability)을 통해 구함
 - 점유 확률 OCC = $(255 - \text{color_avg}) / 255.0$ 로 표현
 - 만약, 이미지가 24bit 면 $\text{color_avg} = (\text{한 셀의 그레이스케일 값} / 0xFFFF \times 255)$
 - OCC 가 1에 가까울수록 점유되었을 확률이 높아지고 0에 가까울수록 비점유 됨을 의미
- ROS의 메시지(nav_msgs/OccupancyGrid)로 발행될 때
 - 점유도를 0 ~ 100 까지로 재정의하여 표현
 - 0에 가까울수록 이동 가능한 자유 영역(free area), 100에 가까울수록 이동 불가능한 점유 영역(occupied area)
 - "-1"은 특별히 확인되지 않은 미지 영역(unknown area) 으로 정의



6. SLAM (Simultaneous Localization and Mapping)

: Map

- ROS에서 지도 정보를 *.pgm 파일 형태(**portable graymap format**)로 저장.



map.yaml

Open

```
image: /home/yongseok/map.pgm
resolution: 0.050000
origin: [-10.000000, -10.000000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

지도 해상도 meters/pixel 단위. 각 픽셀은 5cm 의미

origin

지도의 원점. 각각 x, y, yaw(Vertical axis)
: 지도의 왼쪽 아래가 x = -10m, y = -10m

negate

: 흑백을 반전

: 각 pixel의 흑백 결정은 점유 확률이 점유 한계(**occupied_thresh**)를

: 넘으면 검은색 이동 불가능한 점유 공간(**occupied area**) 표현

: 자유 한계치(**free_thresh**)보다 작으면 흰색 이동 가능한 자유 공간
(**free area**) 표현

: remote PC에서 새터미널 열고, 현재 작업중인 TOPIC list 확인하기

→ rostopic list -v

6. SLAM (Simultaneous Localization and Mapping)

(5) SLAM node 통신

: SLAM을 통한 map 구현 시,

rqt_graph, rviz 를 통하여

node간의 통신과 message 내용을 확인할 수 있음

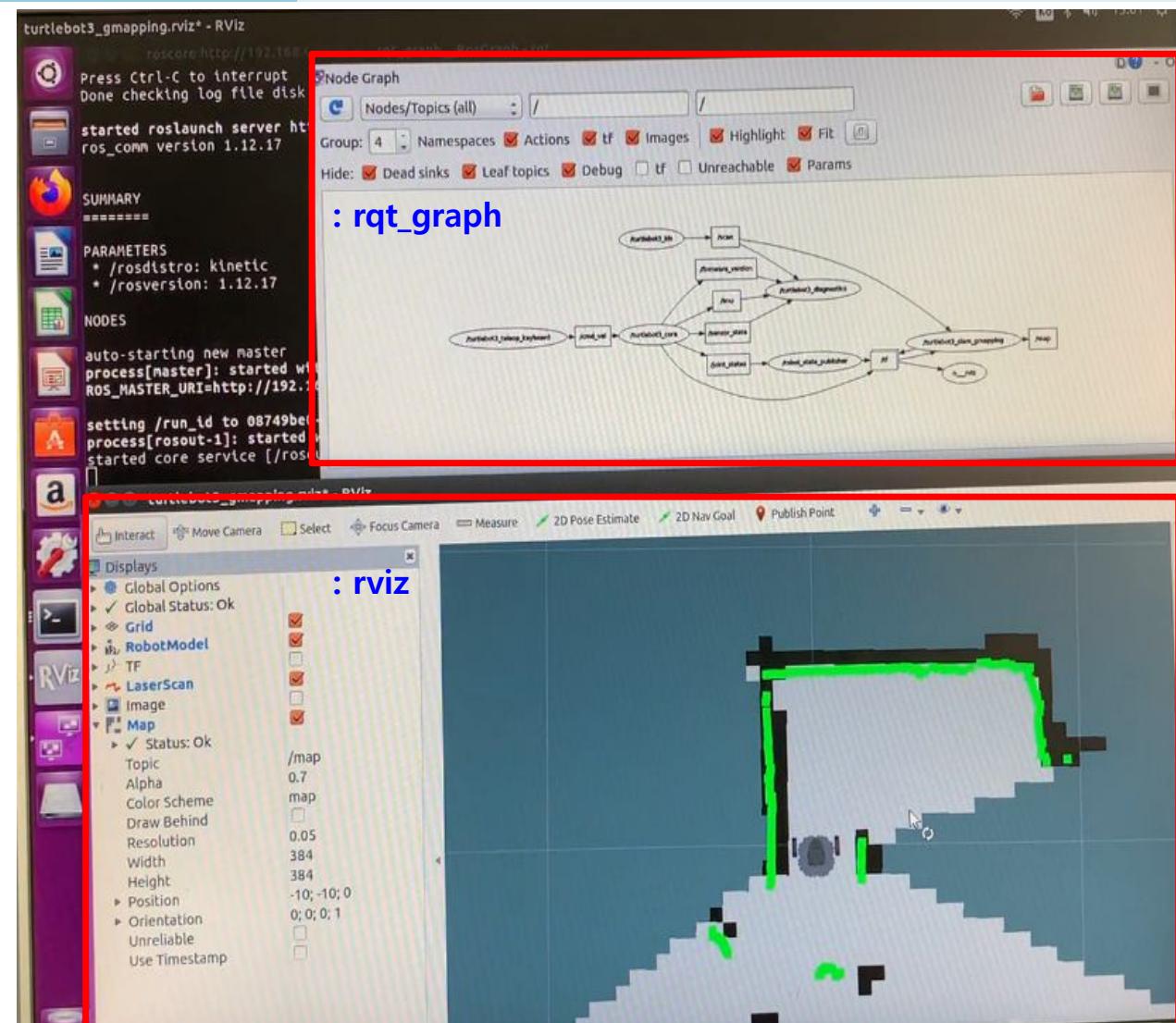
: gmapping 사용한 SLAM node 처리과정

→ remote PC에서 새터미널 열고,

현재 작업중인 TOPIC list 확인하기

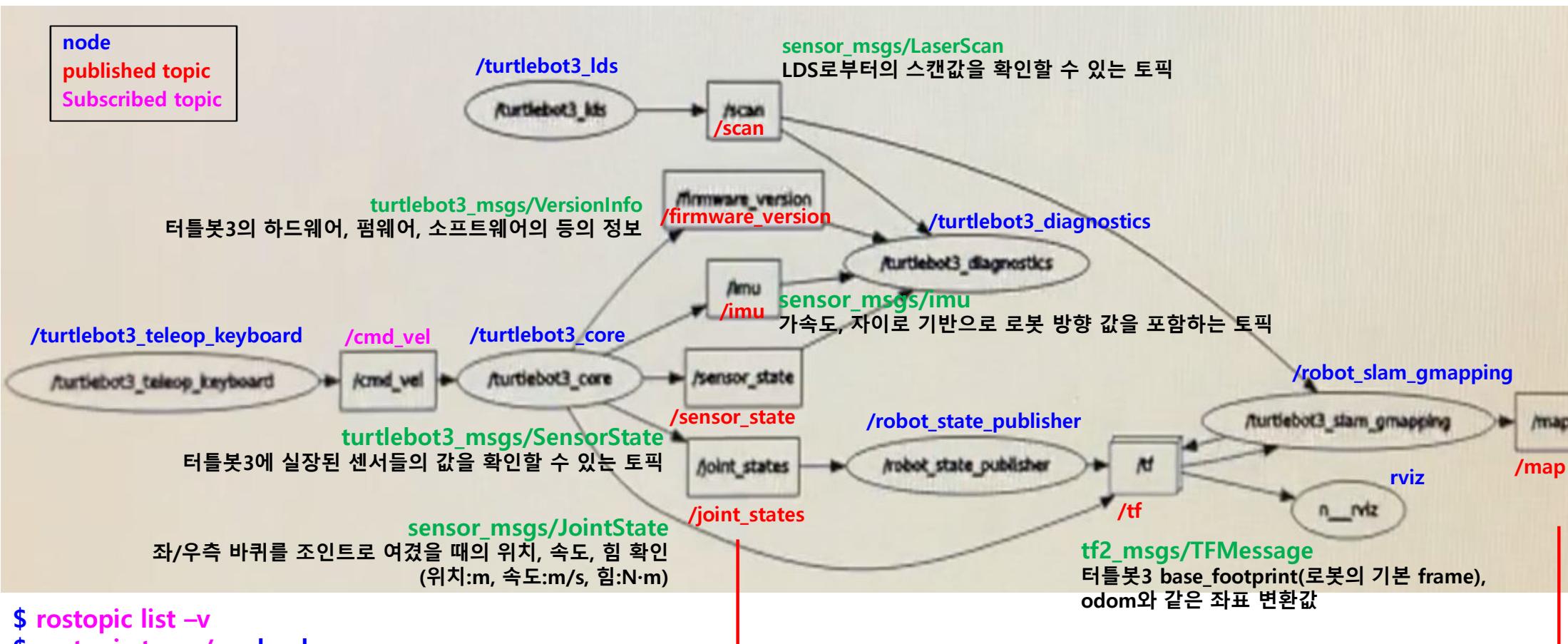
\$ rostopic list -v

→ publish topic, subscribe topic 구분



6. SLAM (Simultaneous Localization and Mapping)

: gmapping 사용한 SLAM node 처리과정 → node간의 통신과 message 내용을 확인



\$ rostopic list -v

\$ rostopic type /cmd_vel

\$ rosmsg show geometry_msgs/Twist

중요

6. SLAM (Simultaneous Localization and Mapping)

: published topic

topic name	topic type	function
/sensor_state	turtlebot3_msgs/SensorState	터틀봇3에 실장된 센서들의 값을 확인할 수 있는 토픽
/scan	sensor_msgs/LaserScan	LDS로부터의 스캔값을 확인할 수 있는 토픽(거리 값 의미)
/imu	sensor_msgs/Imu	가속도, 자이로 센서 값을 기반으로 로봇의 방향 값을 포함하는 토픽
/odom	nav_msgs/Odometry	encoder와 IMU 기반으로 터틀봇3의 Odometry 정보를 얻음 (센서의 위치 같은 로봇의 이동량인 odometry에 의존)
/tf	tf2_msgs/TFMessage	터틀봇3 base_footprint(로봇의 기본 frame), odom와 같은 좌표 변환값 (tf: transform. 자세(위치+방향) 정보는 상대좌표 관계에 따라 바뀌기 때문에 transform이라 표현)

→ 로봇의 움직임에 따른 상대 좌표 base_footprint(로봇의 최상단점), base_link(로봇의 중심점), base_scan(센서의 위치)를 구하고,

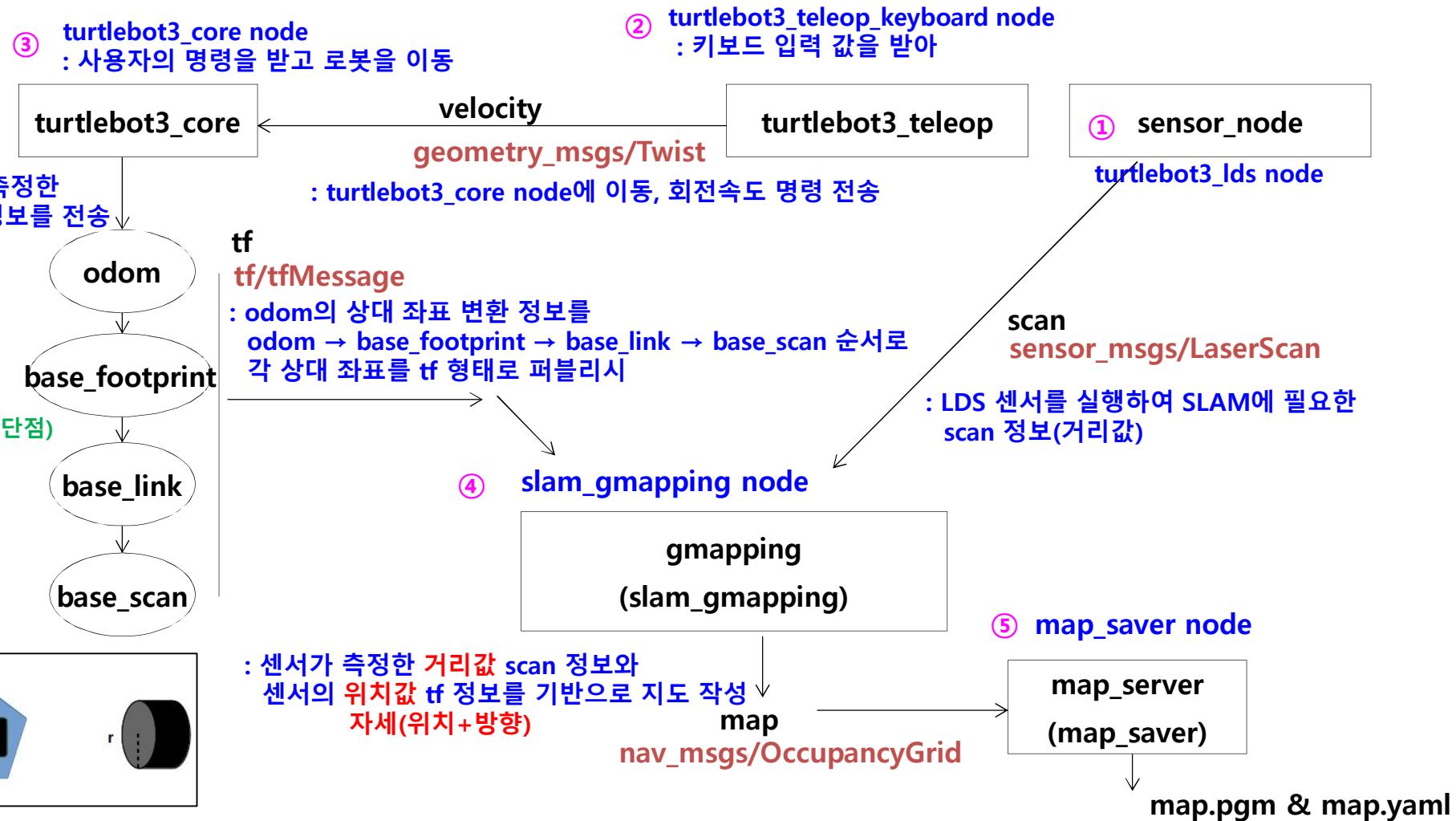
이 값을 tf message gmapping node에 전달

/joint_states	sensor_msgs/JointState	좌/우측 바퀴를 조인트로 여겼을 때의 위치, 속도, 힘 확인 (위치:m, 속도:m/s, 힘:N·m)
/diagnostics	diagnostic_msgs/DiagnosticArray	자기 진단 정보
/firmware_version	turtlebot3_msgs/VersionInfo	터틀봇3의 하드웨어, 펌웨어, 소프트웨어의 등의 정보

6. SLAM (Simultaneous Localization and Mapping)

Ref: ROS 로봇 프로그래밍 Ruby paper

: SLAM node 처리과정



7. Navigation & SLAM (Simultaneous Localization and Mapping)

7-2. SLAM (Simultaneous Localization and Mapping) & Navigation

(1) tf(transform)

: SLAM에 사용되는 정보는 거리 값과 그 거리 값을 계측한 위치가 어느 곳인지 판단

- 거리 값은 센서 node로부터 해당 정보 값을 받을 수 있고,
- 그 거리 값을 계측한 위치는 해당 센서가 어디에 있는지? 센서는 로봇의 어디에 부착되어 있는지?
- robot은 원격 조종 명령(turtlebot3_teleop_key.launch)에 따라 움직임

: robot과 sensor는 물리적으로 고정되어 있고 로봇의 움직임에 따라 상대적으로 sensor의 자세(위치+방향)가 변함

→ 이것을 상대 좌표 변환이라고 하며, ROS는 그 과정을 tf(transform)이라 함

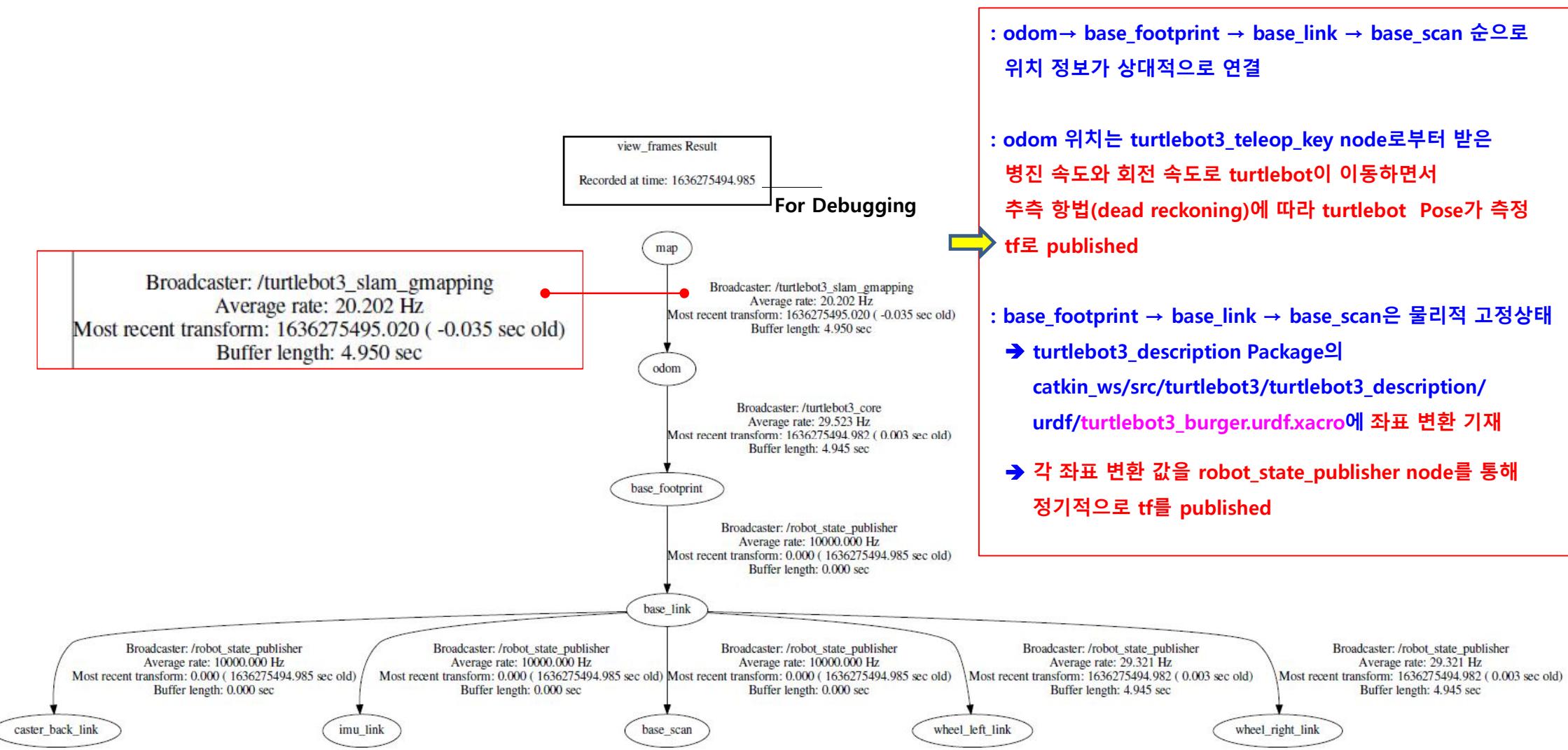
→ robot의 움직임에 따른

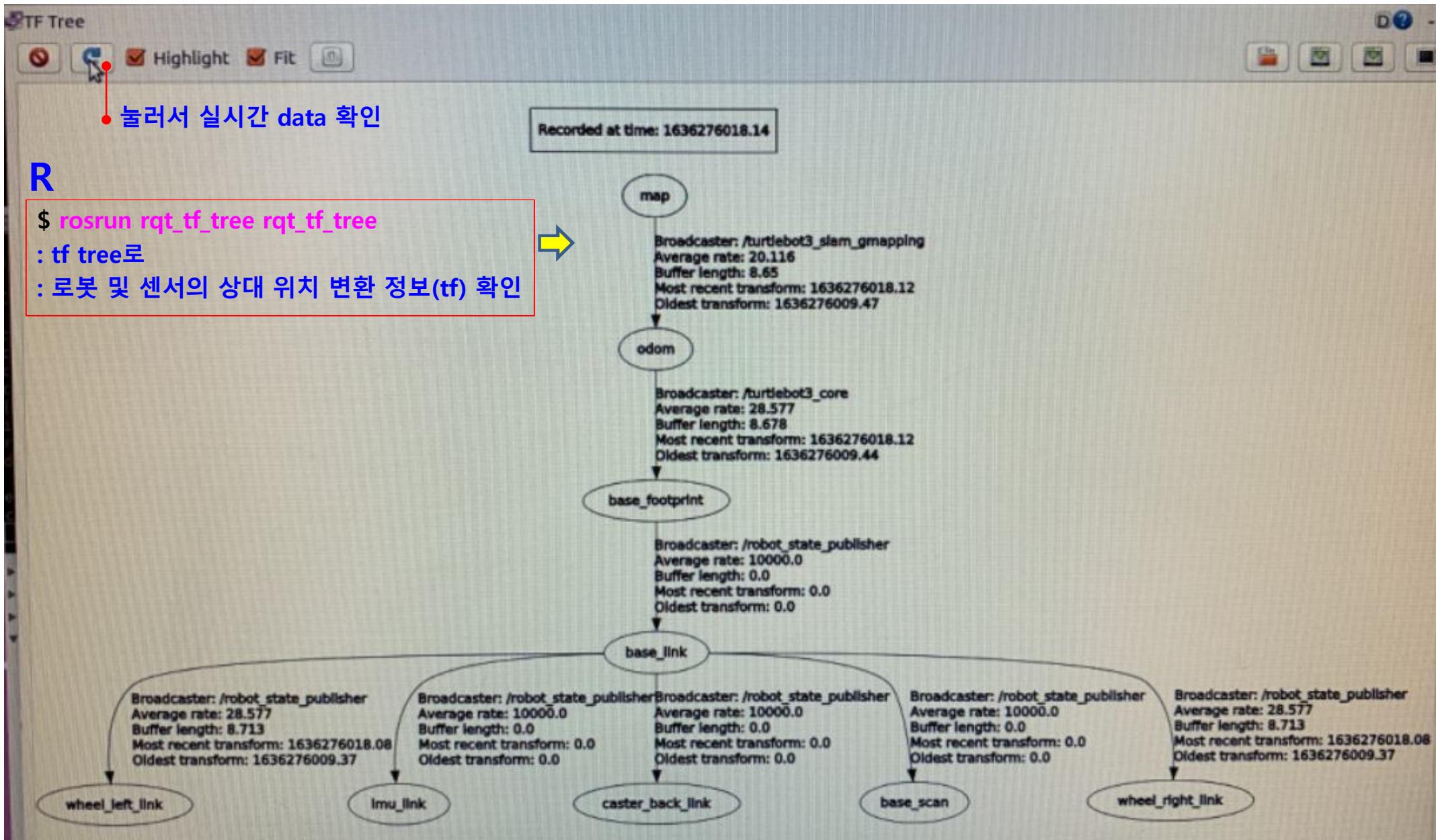
상대 좌표 base_footprint(로봇의 최상단점), base_link(로봇의 중심점), base_scan(센서의 위치)를 구하고,

이 값을 tf message gmapping node에 전달

: tf tool 사용

SLAM시 작은 초록색 화살표 라고 부르기도 함



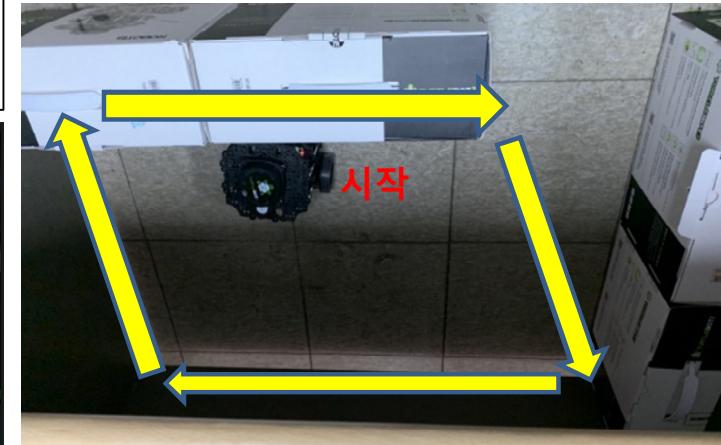


7. Navigation & SLAM (Simultaneous Localization and Mapping)

- ④ turtlebot3_teleop_key.launch 실행하여 아래 그림과 같이 turtlebot 동작시켜 map 생성. 이때 Rviz 화면 확인

R

```
새 터미널 열고 $ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```



- ⑤ 맵 구현이 완료되면 아래와 같이 저장함

R

```
새 터미널 열고 $ rosrun map_server map_saver -f ~/map
```

- ⑥ Rviz 닫고, Navigation 실행 가기, 실행하면 Rviz 화면이 다시 열림(map 파일 불러옴)

R

```
새 터미널 열고 $ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```

7. Navigation & SLAM (Simultaneous Localization and Mapping)

Kinetic SKIP

7-3. SLAM (Simultaneous Localization and Mapping) & Navigation 실습

(1) SLAM을 위한 공간 만들기

Case1.



Case2.



Case3.



: ROS 종속 Package 설치, Ubuntu 환경, SLAM 종류(**gmapping**, **cartographer**, **hector mapping**)에 따라서
SLAM후 Navigation 진행 시 rviz costmap에서 “No map received”라는 warn과 함께 navigation 진행이 되지 않음

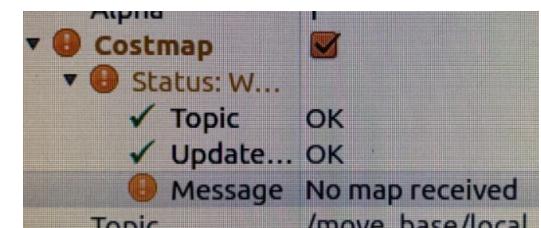
→ 해결방법1. ROS 종속 package 재 설치후, build



→ 해결방법2. 실험 map size(위 그림처럼) 확장 ... 근본적 해결책이 아님

→ 해결방법3. inflation radius 값 변경(작게) 수정

`home/catkin_ws/src/turtlebot3/turtlebot3_navigation/param/costmap_common_param_burger.yaml`



→ 해결방법4. SLAM gmapping 문제발생 경우 SLAM hector mapping로 변경

7. Navigation & SLAM (Simultaneous Localization and Mapping)

Kinetic SKIP

→ 해결방법. 실험 map size(위 그림처럼) 확장 ... 근본적 해결책이 아님

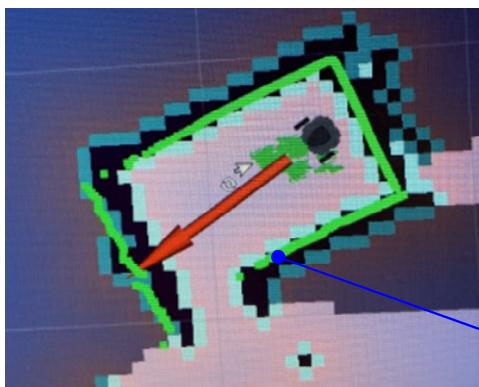
SLAM후 Navigation 진행 시 rviz costmap에서 “No map received”라는 warn과 함께 navigation 진행이 되지 않음

→ rviz, 2d nav Goal 이 동작하지 않는 원인은 robot_status를 제대로 불러오지 못했거나, path 설정을 실패 등 원인.★

Case1.



Rviz : Navigation



Case2.



Case3.



navigation 진행이 되지 않음

7. Navigation & SLAM (Simultaneous Localization and Mapping)

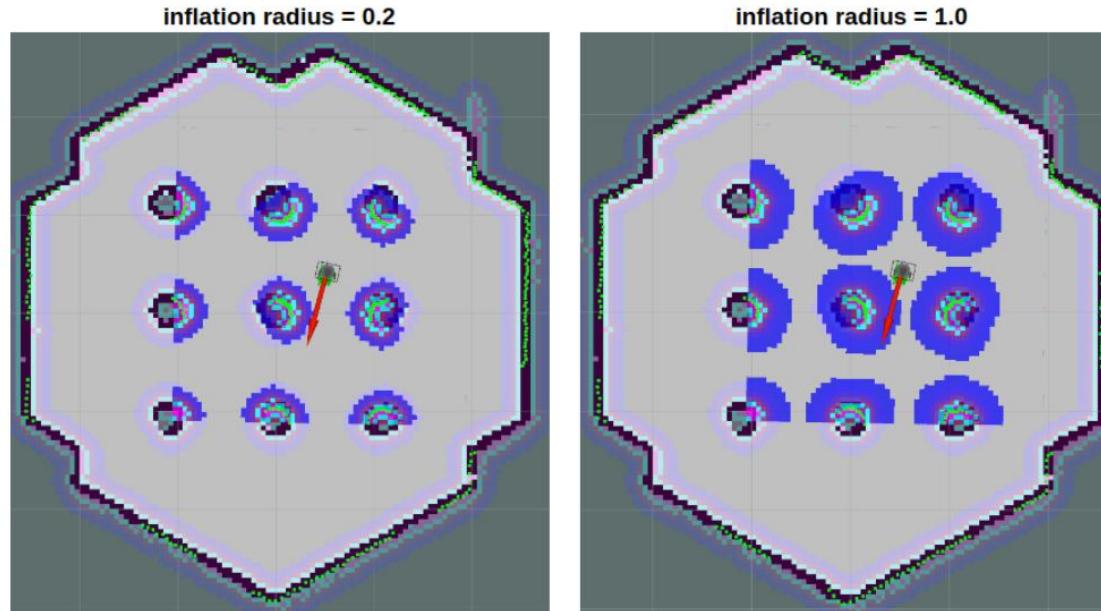
Kinetic SKIP

→ 해결방법. inflation radius 값 변경(작게) 수정

home/catkin_ws/src/turtlebot3/turtlebot3_navigation/param/costmap_common_param_burger.yaml

: parameter에서 장애물 inflation radius(팽창 영역) 설정.

: path plan 설정시 inflation radius을 가로 지를 수 없음 (robot이 움직일 때 충돌 방지를 위함)



Ref. <https://emanual.robotis.com/docs/en/platform/turtlebot3/navigation/#navigation>

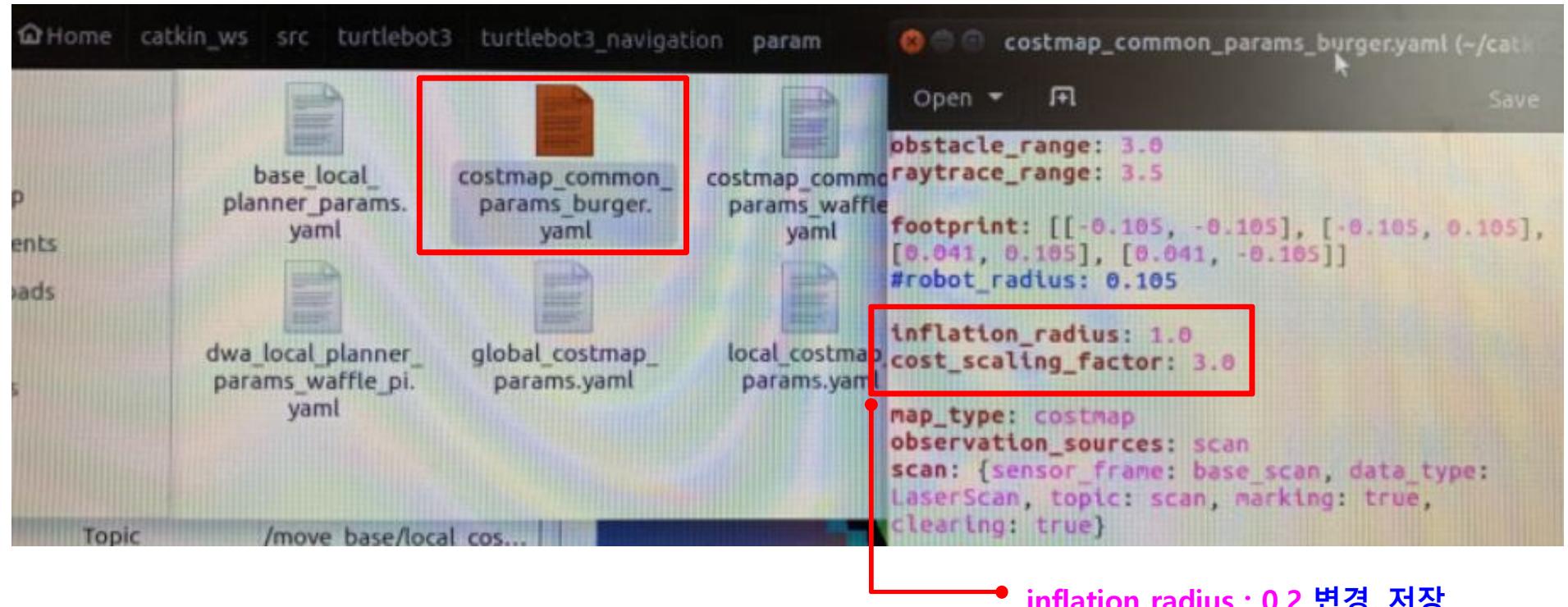
Inflation is the process of propagating cost values out from occupied cells(로봇 이동 불가능한 점유 영역)

7. Navigation & SLAM (Simultaneous Localization and Mapping)

Kinetic SKIP

→ 해결방법. inflation radius 값 변경(작게) 수정

home/catkin_ws/src/turtlebot3/turtlebot3_navigation/param/costmap_common_param_burger.yaml



inflation radius : 0.2 변경, 저장,
그리고 반드시 재부팅

Kinetic SKIP

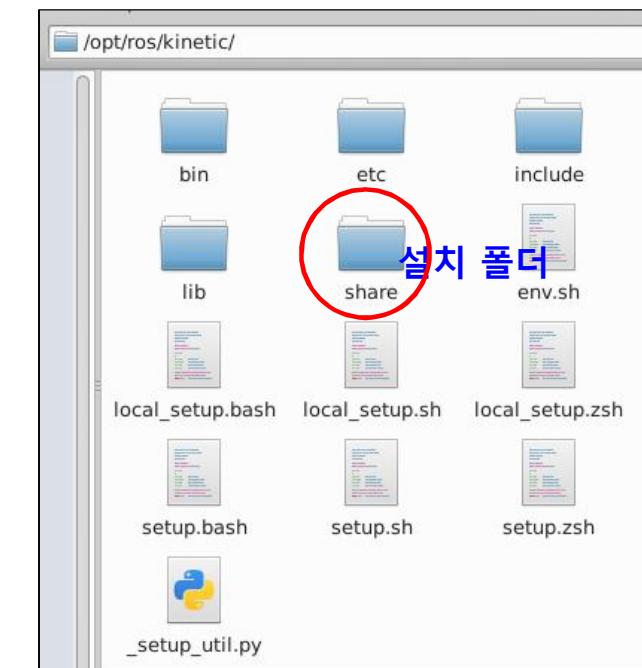
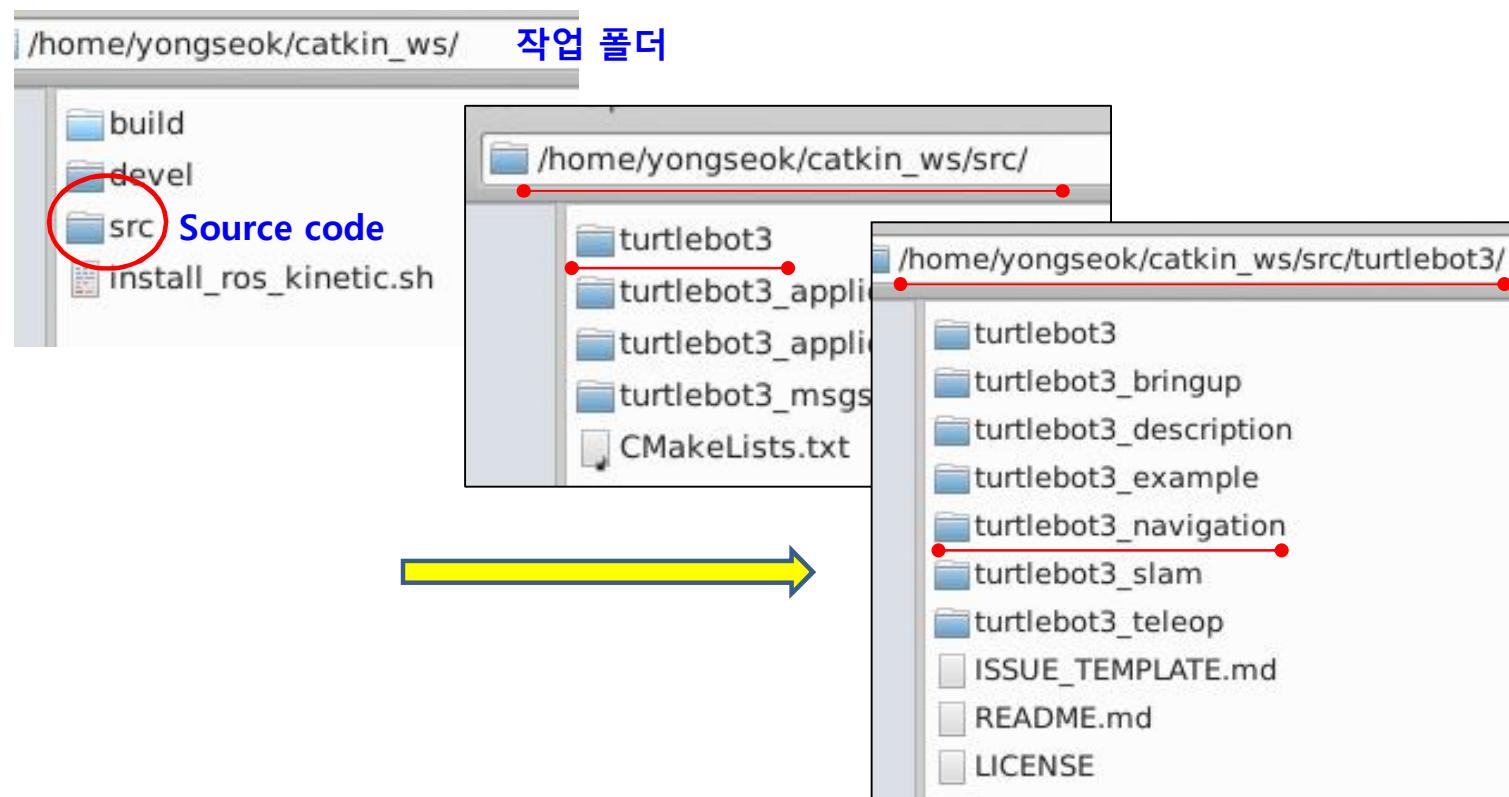
: ROS 파일 시스템은 설치 폴더와 사용자 작업 폴더로 구분

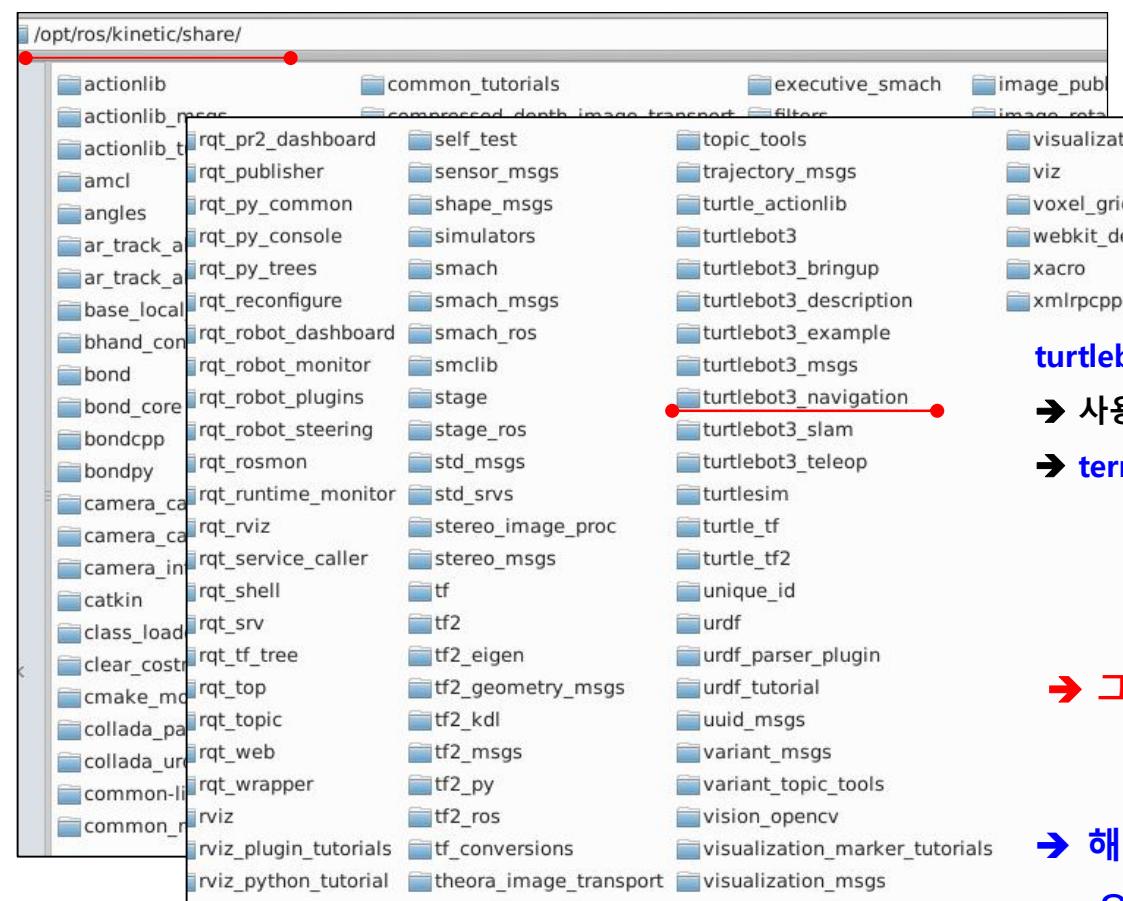
- ROS 설치 폴더는 ROS를 설치하면 /opt 폴더에 ros 폴더 생성

→ ROS 설치 폴더 경로 /opt/ros/kinetic

→ ros 폴더에 roscore를 포함한 핵심 유ти리티와 rqt, Rviz, 로봇 관련 라이브러리, 시뮬레이션, 내비게이션 등 설치

- 사용자 작업 폴더는 사용자 폴더인 ~/catkin_ws 사용





turtlebot3_navigation/param/costmap_common_param_burger.yaml

→ 사용자 폴더가 아니라, opt 폴더(설치폴더)이므로

→ terminal창에서 cd 경로 넣고 위치 변경 후

`sudo chmod 755 costmap_common_param_burger.yaml`

`sudo nano costmap_common_param_burger.yaml`

여기서 수정하고 저장하기

→ 그러나 작업 폴더가 아니므로, inflation radius 적용 안됨



→ 해결방법. inflation radius 값 변경(작게) 수정

오직 작업 폴더인

`home/catkin_ws/src/turtlebot3/turtlebot3_navigation/param`에서

`costmap_common_param_burger.yaml` 수정하기

★
만일 작업 폴더 `home/catkin_ws/src`에 source code등 파일 없으면
ROS 설치 중에, 환경변수 설정이 안되었거나,
오류(path)에 빠졌거나
우분투 설치 문제임 (방법은 우분투부터 재설치)

7. Navigation & SLAM (Simultaneous Localization and Mapping)

Kinetic SKIP

7-3. SLAM (Simultaneous Localization and Mapping) & Navigation 실습

(2) Navigation with SLAM gmapping

ⓐ SLAM을 위한 공간 준비



ⓑ SLAM gmapping으로 map 파일 만들기(PPT61장 참고)

: \$ `ifconfig` RemotePC(Ubuntu)와 SBC(Raspberry Pi, Turtlebot3) id 확인

R 기존의 모든 터미널을 닫고(`roscore` 실행 중의 터미널은 `ctl + c`로 닫고).
새 터미널 열고 \$ `roscore`



R 새 터미널 열고 \$ `ssh pi@192.168.0.21 password is turtlebot`
turtlebot3로 연결됨.

T 즉 Remote PC의 terminal창에 turtlebot3의 terminal이 생성됨
\$ `roslaunch turtlebot3_bringup turtlebot3_robot.launch`
→ LDS 회전 확인

R 새 터미널 열고
\$ `roslaunch turtlebot3_slam turtlebot3_slam.launch`
→ RemotePC에 Rviz 화면이 열림

ⓒ turtlebot 전원을 끄지 말고 바닥에 내려놓기



7. Navigation & SLAM (Simultaneous Localization and Mapping)



R

```
새 터미널 열고 $ roscore
```

R-S

```
새 터미널 열고 $ ssh ubuntu@192.168.0.21 (IP is Raspberry, password turtlebot)
```

```
$ rosrun turtlebot3_bringup turtlebot3_robot.launch
```

→ LDS 회전 확인(회전 안하면)

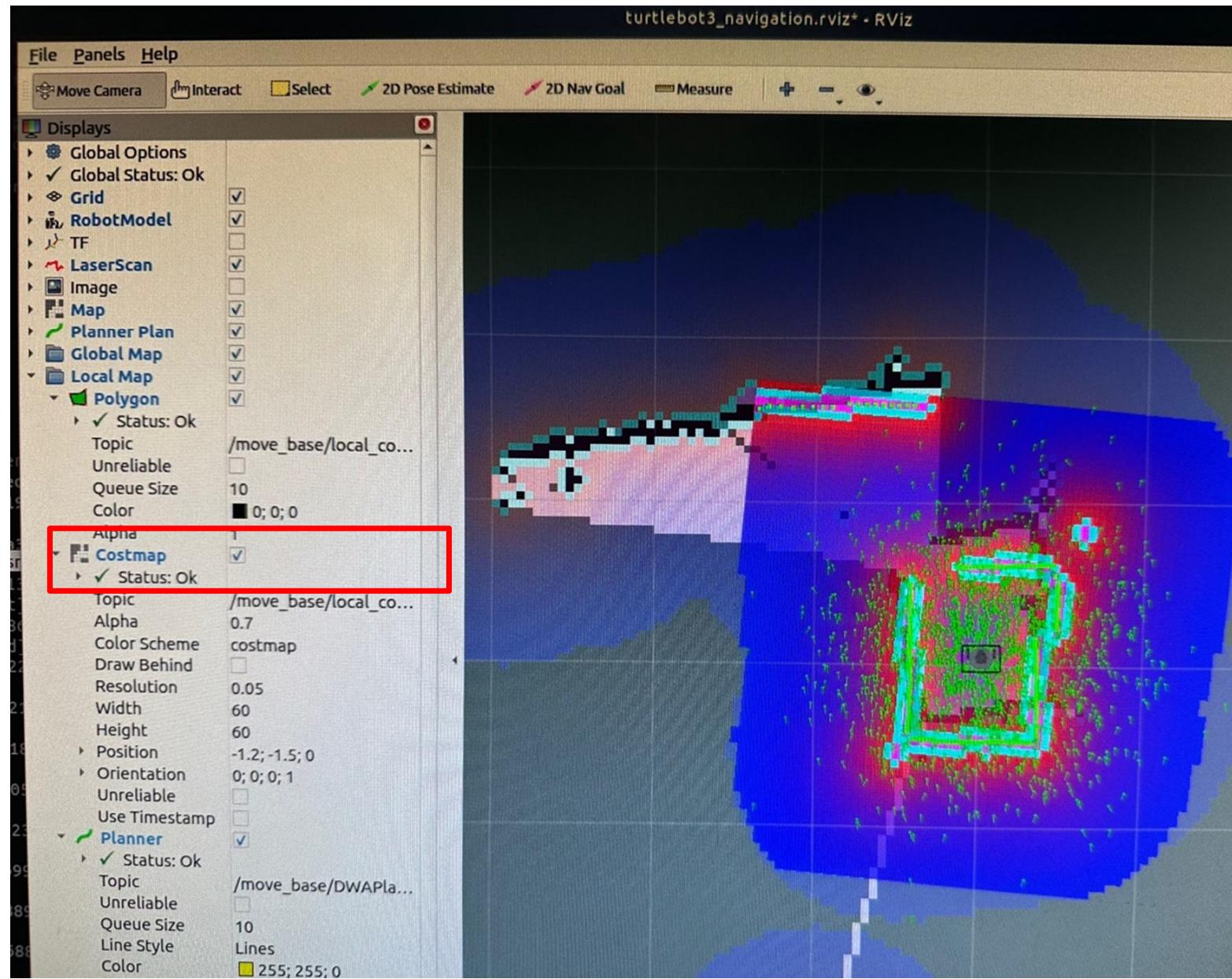
R

```
새 터미널 열고 $ rosrun turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```



RViz 화면이 열림

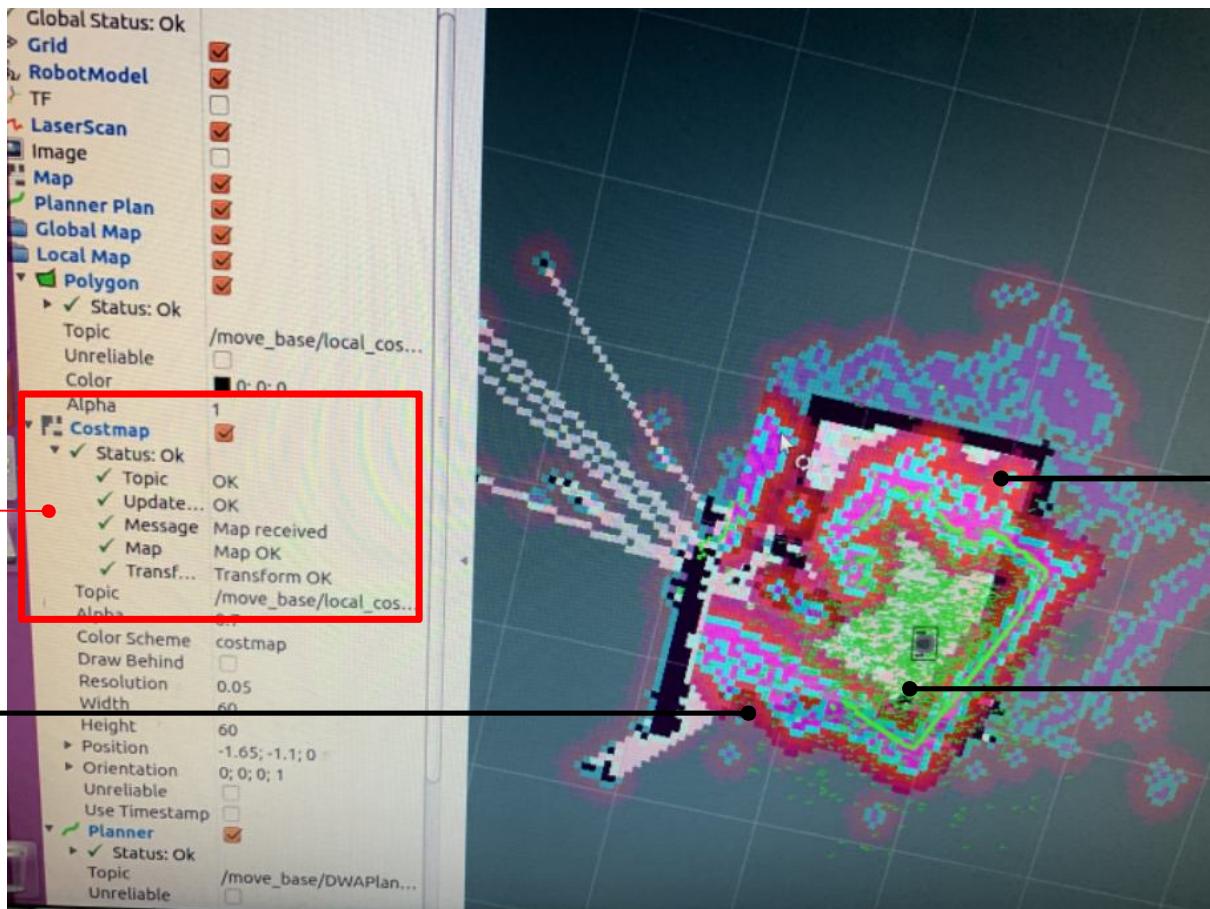
RViz 화면이 열림



7. Navigation & SLAM (Simultaneous Localization and Mapping)

⑨ Rviz 닫고, Navigation 실행 가기, 실행하면 Rviz 화면이 다시 열림(map 파일 불러옴)

```
$ rosrun turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```



- LDS와 자기 인식(Pose)
(Dynamic Window Approach)
- Dynamic Window 정상

Rviz 화면에서, 2D Pose
Estimate 누르고, click화면
큰 화살표가 생기며, drag하
면 화살표가 형성되며, map
과 실제가 점차 일치해 나감
같은 과정을 반복하여, map
과 실제가 일치하도록 함

일치될 수록 초록색
화살표가 turtlebot에
집중됨

7. Navigation & SLAM (Simultaneous Localization and Mapping)

④ 이때 map과 일치가 잘되지 않을 경우에는 `turtlebot3_teleop_key.launch` 실행하여 turtlebot 동작시켜(회전) map 일치시킴

R

새 터미널 열고 \$ `roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch`

Map과 일치되면 반드시,

turtlebot3_teleop_key.launch를
CTL + C 중단 후 터미널을 닫아야 함
: `turtlebot3_teleop_key.launch` 실행되면
turtlebot은 기본적으로 정지상태



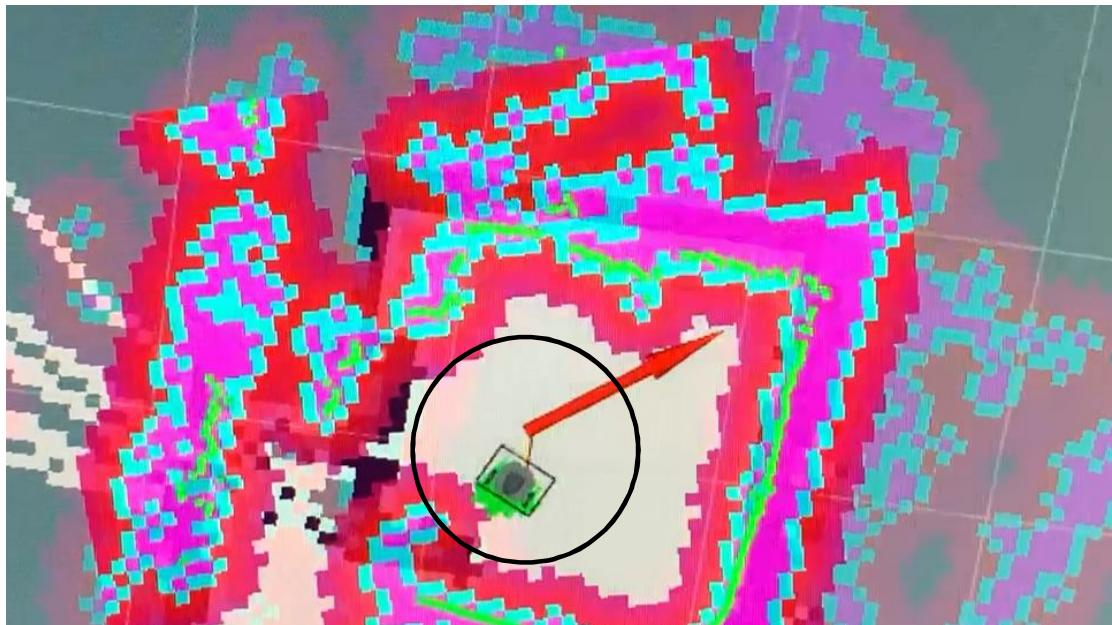
Terminator에서,
`turtlebot3_teleop_key.launch`
현재 활성화 된 창 닫기 : Ctrl + Shift + W

일치될 수록 초록색
화살표가 turtlebot에
집중됨

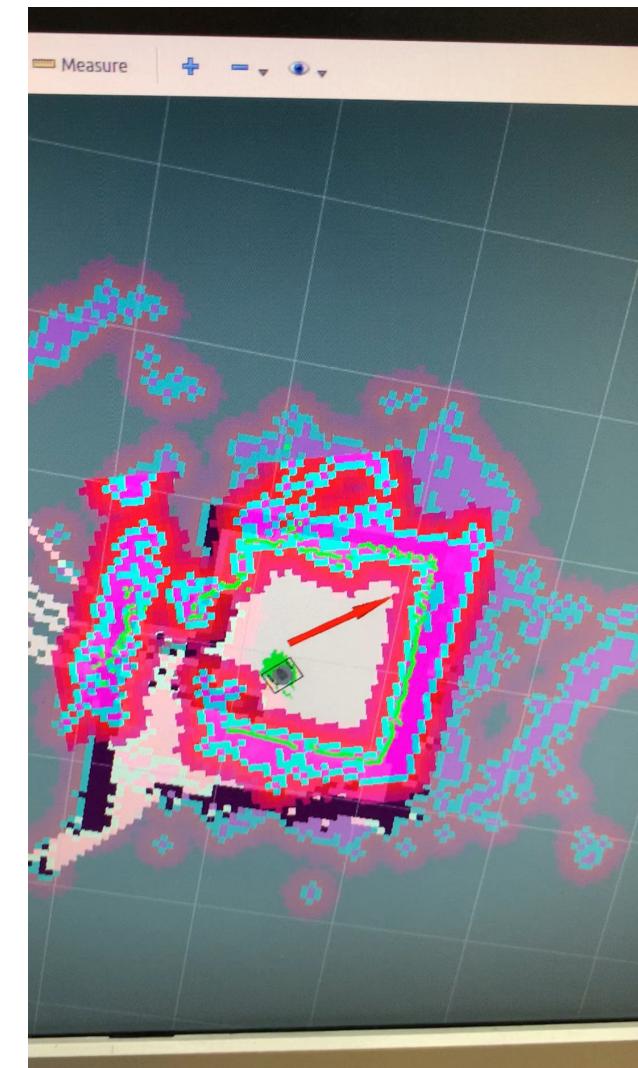
7. Navigation & SLAM (Simultaneous Localization and Mapping)

① map 일치되면,

Rviz 화면에서, 2D Nav Goal 누르고, Turtlebot 정면을 click하면 큰 화살표(초록색)가 생기며,
목표하는 곳으로 drag하면 초록 화살표가 형성되고,
mouse에서 손을 띠면 빨간색 화살표 생기며
turtlebot 자동으로 이동하게 됨. 반복하여 구동 가능



동영상
→



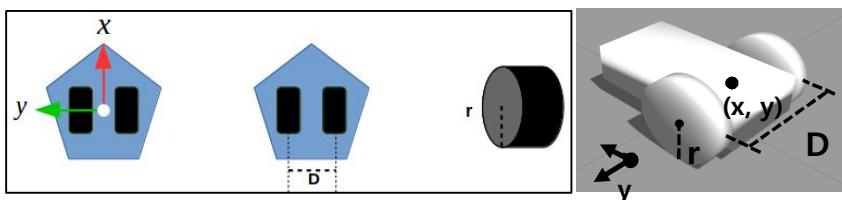
7. Navigation & SLAM (Simultaneous Localization and Mapping)

7-4. SLAM 위치 추정(Localization) 알고리즘

(1) 위치 추정

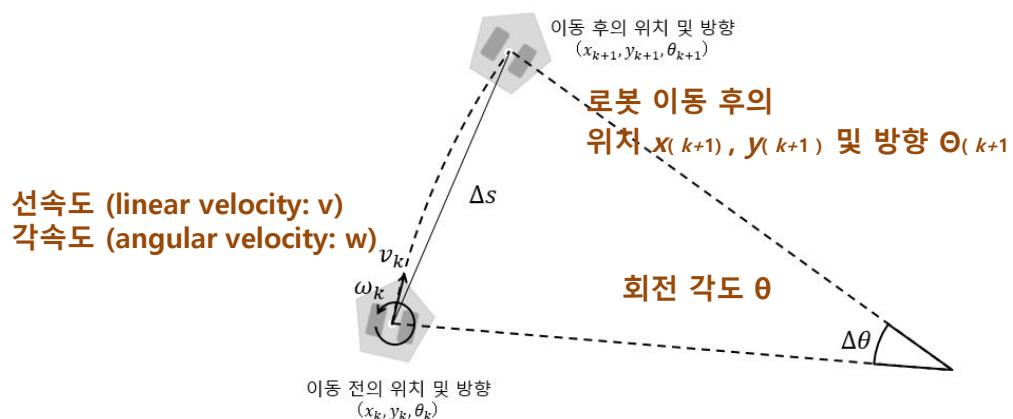
추측 항법(dead reckoning)

- 양 바퀴 축의 회전 값을 이용(encoder: wheel의 회전 량 측정)
- 이동 거리와 회전 값을 계산, 위치 측정



바퀴 간 거리 D , 바퀴 반지름 r

- IMU(자이로/가속도/지자계) 등 관성 센서 및 필터로 위치 보상



SLAM
위치 추정

센서 관측 정보 불확실
실제 환경에서 실시간성 확보 필요



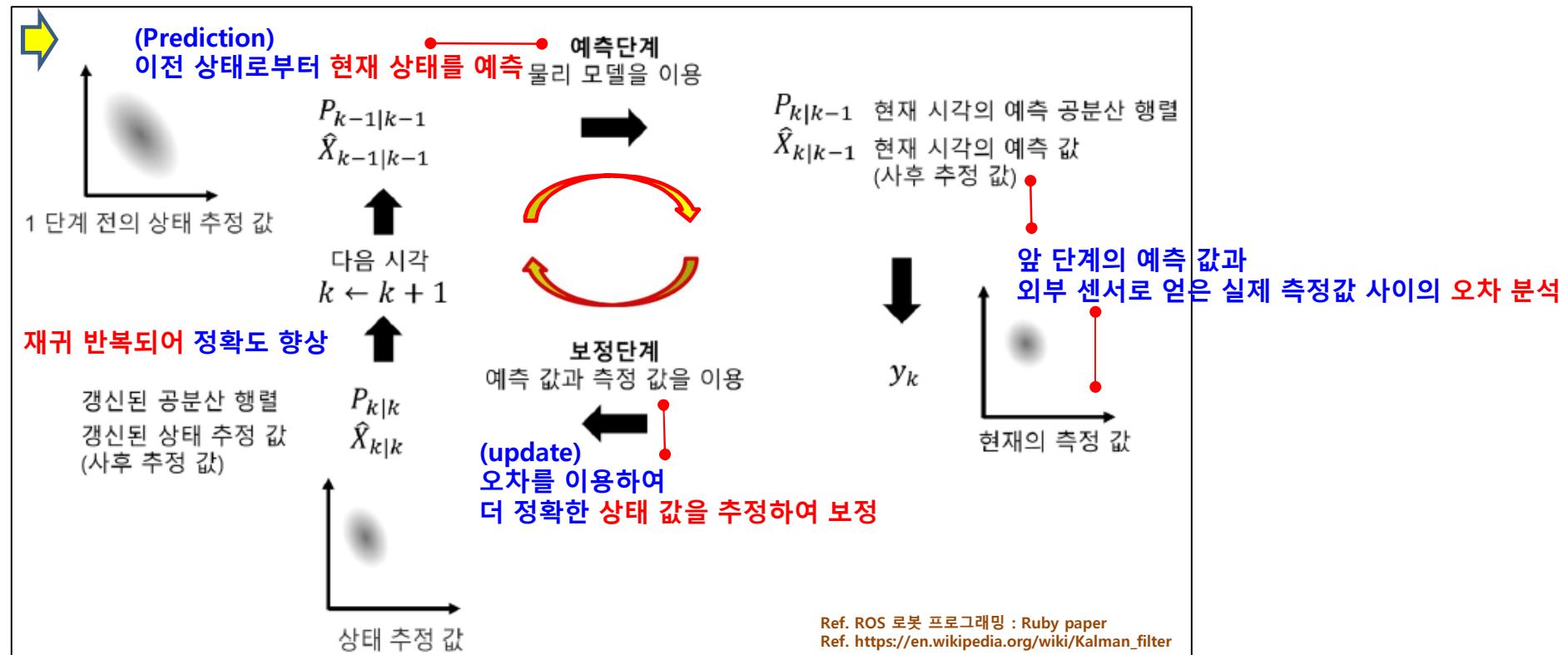
Kalman filter, Particle filter

7. Navigation & SLAM (Simultaneous Localization and Mapping)

7-4. SLAM 위치 추정(Localization) 알고리즘

(2) Kalman filter

- NASA의 아폴로 project에 사용된 Rudolf E.Kalman 박사가 개발
- Noise(Gaussian Noise)가 포함된 선형 시스템(linear system)에서 object의 상태를 추적하는 재귀 필터(recursive filter)



7. Navigation & SLAM (Simultaneous Localization and Mapping)

(2) Kalman filter

- 수식 유도 과정 (http://jinyongjeong.github.io/2017/02/14/lec03_kalman_filter_and_EKF/)
- Object를 선형시스템으로 가정하고 선형 운동으로 최적의 parameter를 찾아가는 해석적 방법
- Kalman filter는 선형 시스템(linear system)에만 적용
 - Linear system과 Gaussian Noise 적용된 시스템에서만 정확도 보장
- 로봇과 센서는 대부분 비선형 시스템
 - Kalman filter는 정확도가 보장되지 않음
 - Kalman filter를 수정 확장한 EKF(Extended Kalman Filter) 사용
 - : motion model(target이 어떻게 움직일지 예측, path planning)
 - : observation model을 비선형 함수로 확장한 것
 - EKF의 정확성 보완한 무향 칼만 필터 UKF(Unscented Kalman Filter)
 - 속도 개선한 Fast Kalman filter

7. Navigation & SLAM (Simultaneous Localization and Mapping)

(3) Particle Filter

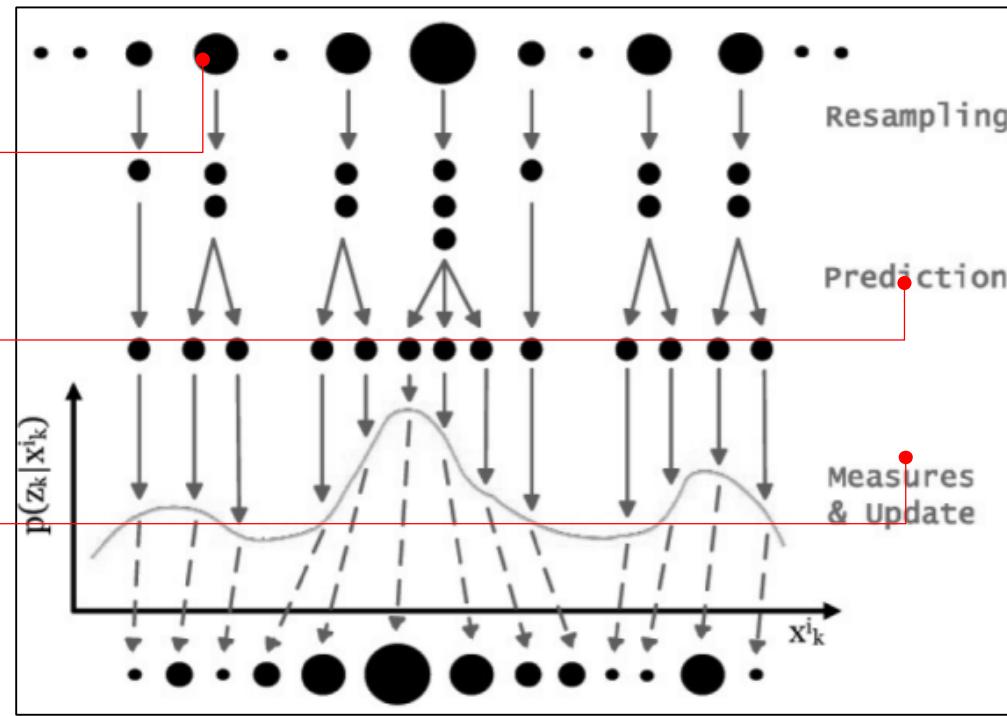
- 물체 추정에 최근 가장 많이 사용되는 알고리즘
: Noise가 있는 환경에서 측정된 data를 filter로 사용해 실제 위치를 추정하는 방법
- 시행착오(try and error)법에 기반한 simulation을 통한 예측 기술
: 여러가지 방법을 실험해, 결과들의 인과관계를 추론하여 문제를 해결하는 방식
- 대상 시스템에 확률 분포로 임의 생성된 추정 값을 particle(입자) 형태로 나타내어 Particle Filter 라고 함
→ SMC(Sequential Monte Carlo) 또는 Monte Carlo Localization 방법이라 함
- 연속해서 들어오는 정보 중에 오차가 포함되었다고 가정하고 object의 위치를 추정
→ SLAM에서 robot의 odometry 값과 LDS를 활용, 관측 값으로 사용되어 robot의 현재 위치 추정

7. Navigation & SLAM (Simultaneous Localization and Mapping)

(3) Particle Filter

- 위치 불확실성을 sample이라 부르는 Particle(입자)로 묘사
: 그림의 검은 점들(크기는 weight의 크고 작음)
- Particle을 로봇의 motion model과 확률에 근거하여
새로운 추정 위치와 방향으로 이동
- 실제 계측 값에 따라 각 입자에 가중치(weight)를 주며
- 점차적으로 정확한 위치로 잡음을 줄이며 추정해나가는 과정
- 각 Particle(입자)을 **particle=pose(x, y, i), weight**로 나타냄
: 각 Particle(입자)은 로봇의 추정 위치와 방향을 나타내는 임의의 작은 Particle로,
로봇의 x, y, i 그리고 각 Particle의 가중치(weight)로 표현

Ref. Real-Time Tracking of Multiple Moving Objects Using Particle Filters and Probabilistic Data Association (ISSN 0005-1144)



Particle Filter의 estimation cycle

(3) Particle Filter



- Particle Filter의 estimation cycle

① 초기화(initialization)

- : 초기 robot의 pose(위치, 방향)를 전혀 알 수 없음
 - n개의 Particle로 자세(pose)를 구할 수 있는 범위 내에서 무작위 배치
 - n개의 Particle weight는 $1/n$ 로, 합은 1 (n은 경험적으로 결정하고, 일반적으로 수백 개)
- : 초기 robot의 pose가 알려진 경우에 그 근처로 Particle 재배치

② 예측(prediction)

- : robot의 움직임을 system model에 기초하여
- : robot Odometry 정보 등을 통해 관측된 이동량에 noise을 포함하여 각 Particle들을 이동시킴

③ 보정(update)

- : 계측된 센서 정보들을 기반으로 각 Particle 존재할 확률 계산하고, 이를 반영하여 각 Particle 가중치(weight) 갱신

④ 자세 추정(pose estimation = localization)

- : 모든 Particle의 위치, 방향, 가중치를 이용하여
 - 가중치 평균, 중앙값, 최대 가중치를 갖는 Particle 값 등으로 robot의 추정 자세 계산

⑤ 재추출(Resampling)

- : 새로운 Particle를 생성하는 단계
- : 가중치가 작은 Particle를 제거하고 가중치가 높은 Particle를 중심으로
 - 기존 Particle 특성인 자세 정보를 갖는 새로운 Particle를 추가로 생성 (Particle 수 n은 유지)

7-5. Navigation : turtlebot3_navigation.launch

The screenshot shows a terminal window with the following details:

- Path: /home/yongseok/catkin_ws/src/turtlebot3/turtlebot3_navigation/launch/
- File List:
 - amcl.launch
 - move_base.launch
 - turtlebot3_navigation.launch
- File Content (turtlebot3_navigation.launch):

```
<launch>
  <!-- Arguments -->
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
  <arg name="map_file" default="$(find turtlebot3_navigation)/maps/map.yaml"/>
  <arg name="open_rviz" default="true"/>
  <arg name="move_forward_only" default="false"/>

  <!-- Turtlebot3 -->
  <include file="$(find turtlebot3_bringup)/launch/turtlebot3_remote.launch">
    <arg name="model" value="$(arg model)" />
  </include>

  <!-- Map server -->
  <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)"/>

  <!-- AMCL -->
  <include file="$(find turtlebot3_navigation)/launch/amcl.launch"/>

  <!-- move_base -->
  <include file="$(find turtlebot3_navigation)/launch/move_base.launch">
    <arg name="model" value="$(arg model)" />
    <arg name="move_forward_only" value="$(arg move_forward_only)"/>
  </include>

  <!-- rviz -->
  <group if="$(arg open_rviz)">
    <node pkg="rviz" type="rviz" name="rviz" required="true"
          args="-d $(find turtlebot3_navigation)/rviz/turtlebot3_navigation.rviz"/>
  </group>
</launch>
```

Annotations in blue text are present on the left side of the code:

- AMCL** (Adaptive Monte Carlo Localization)
- move_base**
- : path planning에 필요한 costmap parameters
- : 이동속도 명령을 robot에 넘기는
- dwa_local_planner에 대한 parameters
- : path planning을 총괄하는 move_base의 parameters를 제어

7. Navigation & SLAM (Simultaneous Localization and Mapping)

7-6. Costmap

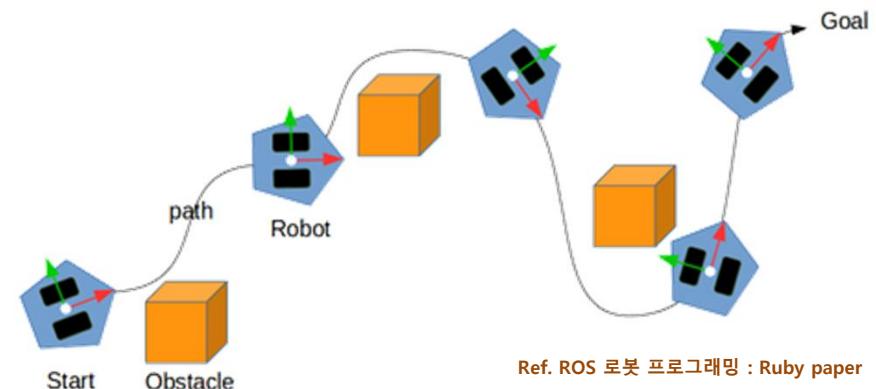
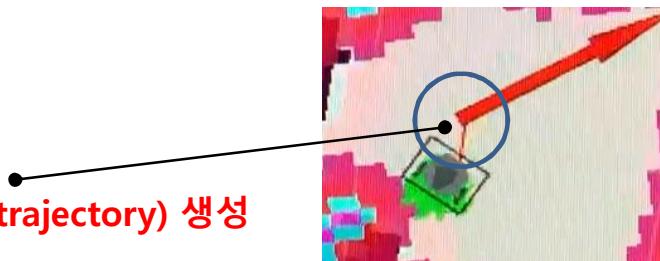
(1) Navigation processing 이해

- ① 위치: 로봇의 위치 계측/추정하는 기능 → **Localization, pose estimation** 라 부름
- ② 센싱: 벽, 물체 등의 장애물의 계측하는 기능 → **sensing**
- ③ 지도: 길과 장애물 정보가 담긴 지도(map 생성)
- ④ 경로: 목적지까지 최적 경로를 계산하고 주행하는 기능
→ **motion planning(path planning)**라고 부름
 - 현재 위치부터 지도상에 지정 받은 목표지점까지 이동 궤적(trajectory) 생성
 - 지도 전체의 전역 이동 경로 계획(global path planning)과
 - robot 중심의 일부 지역으로 국부 이동 경로 계획(local path planning)으로 구분하여 이동 경로 생성

⑤ 장애물 회피(collision avoidance)

: 이동 궤적에 따라 목적지까지 이동중 갑자기 나타난 장애물 회피

→ **Dynamic Window Approach(DWA)** 알고리즘 사용



Ref. ROS 로봇 프로그래밍 : Ruby paper

7. Navigation & SLAM (Simultaneous Localization and Mapping)

7-6. Costmap

(2) Definition

- ① Encoder, 관성 센서(IMU)로부터 얻은 Odometry 정보를 기반으로 robot 위치를 추정
- ② Robot에 장착된 LDS(거리 센서)로 robot과 obstacles(장애물)과의 거리 계산
- ③ Navigation은 robot pose, sensor 자세, obstacles 정보, SLAM의 결과로 얻은 Occupancy Grid Map(점유격자지도)를 고정 지도(static map)로 호출하여, 점유 영역(occupied area), 자유 영역(free area), 미지 영역(unknown area) 정보를 표시



robot pose(위치), sensor 자세, obstacles 정보, SLAM의 결과로 얻은 Occupancy Grid Map



장애물 영역, 장애물과 충돌이 예상되는 영역, robot이 이동 가능한 영역을 계산하며 이것을 “costmap” 라고 함

(3) costmap 종류

- ① costmap : global_costmap, local_costmap 2종류
- ② **global_costmap**
 - : 전역 이동 경로 계획에서 고정 지도의 전체 영역을 대상으로 이동 계획 수립을 위한 지도
- ③ **local_costmap**
 - : 국부 이동 경로 계획에서 robot을 중심으로 한정된 영역에 대해 이동 계획할 때나 장애물 회피 때 사용되는 지도

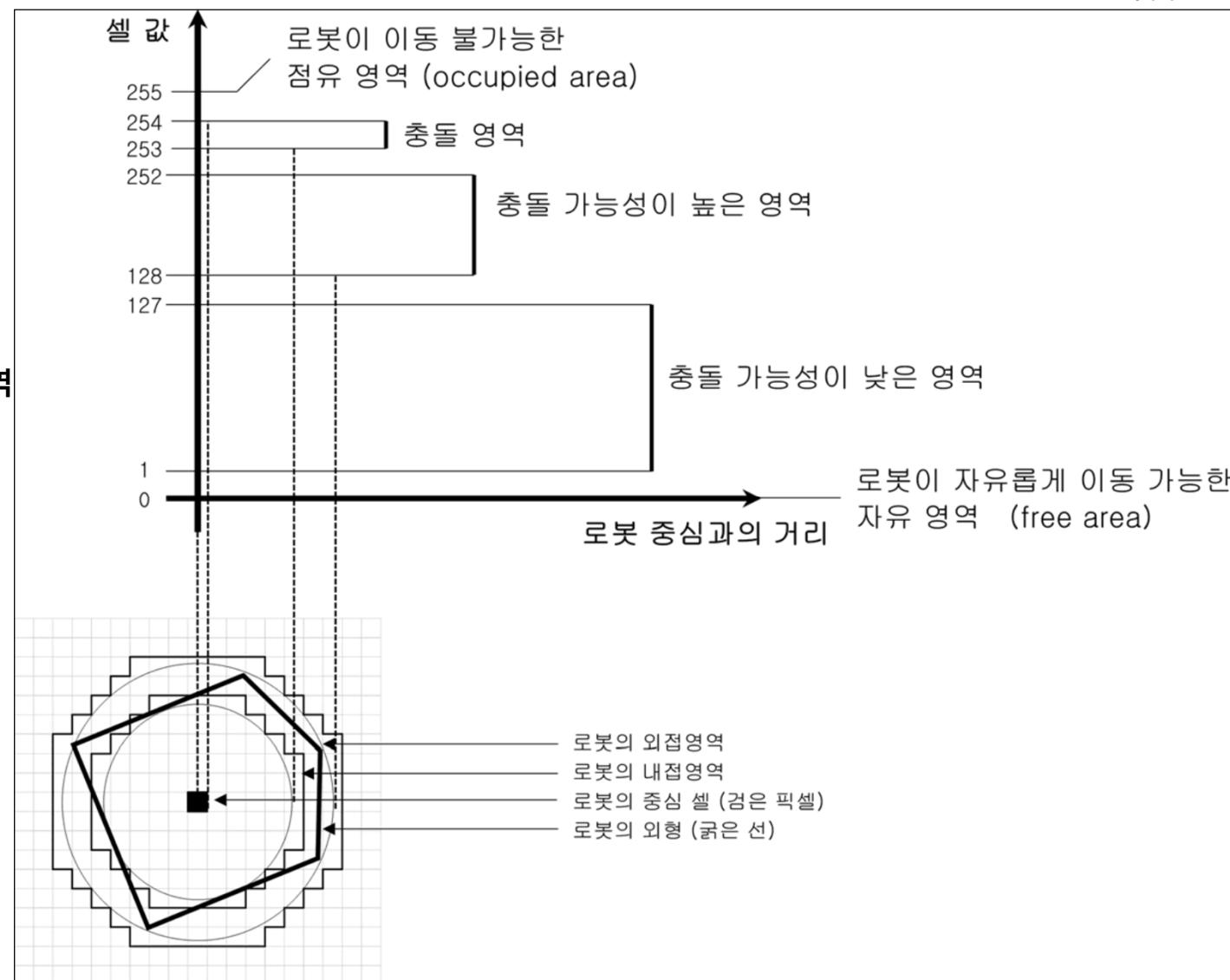
7-6. Costmap

(4) 장애물과의 거리와 costmap 값의 관계도

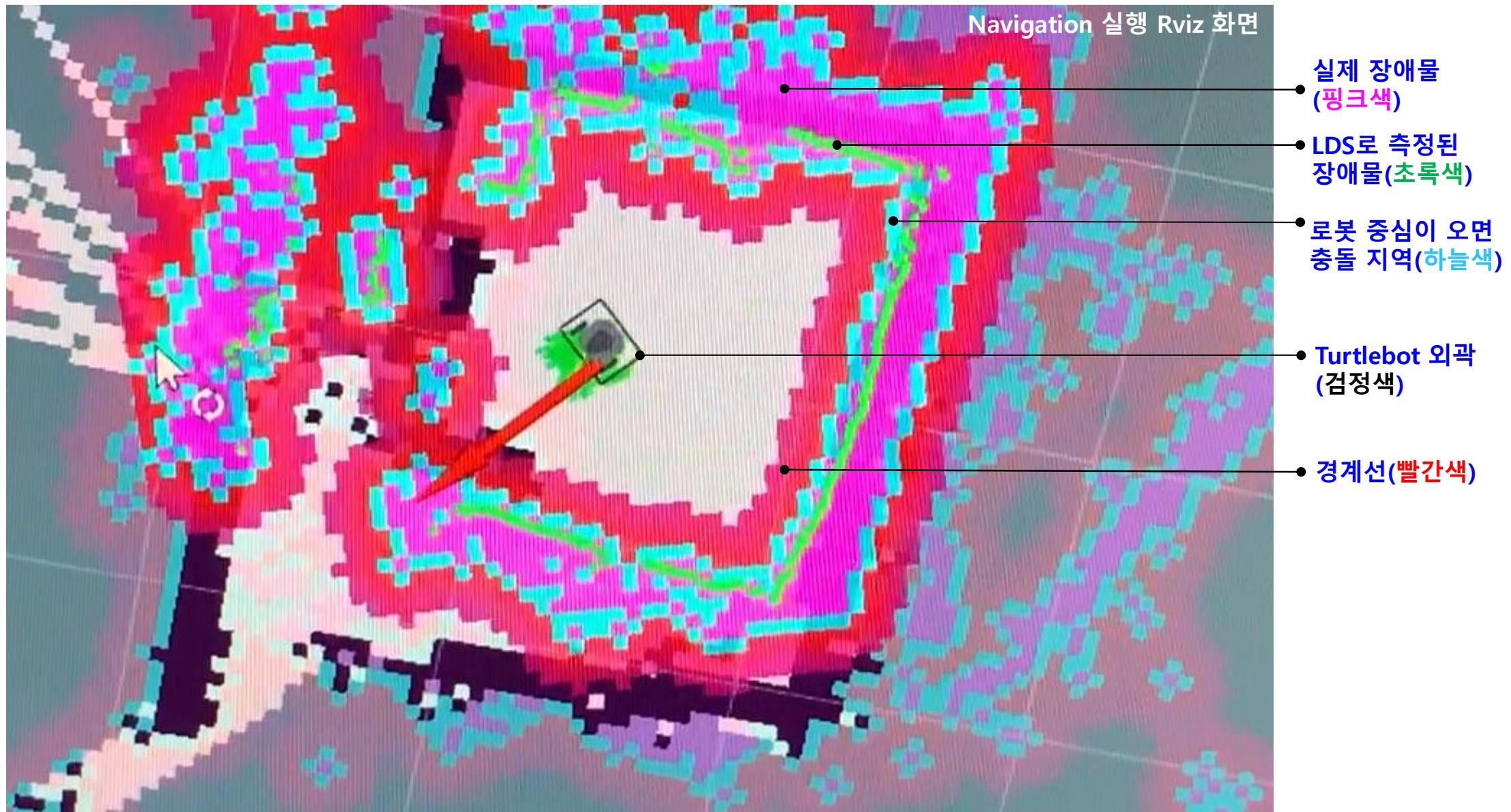
cell cost 값 :

- 255 : 로봇 이동 불가능한 점유 영역
(occupied area)
- 253~254 : 충돌 영역
- 128~252 : 충돌 가능성이 높은 영역
- 001~127 : 충돌 가능성이 낮은 영역
- 000 : 로봇 자유롭게 이동 가능한 자유 영역
(free area)

Ref. ROS 로봇 프로그래밍 (Ruby paper)

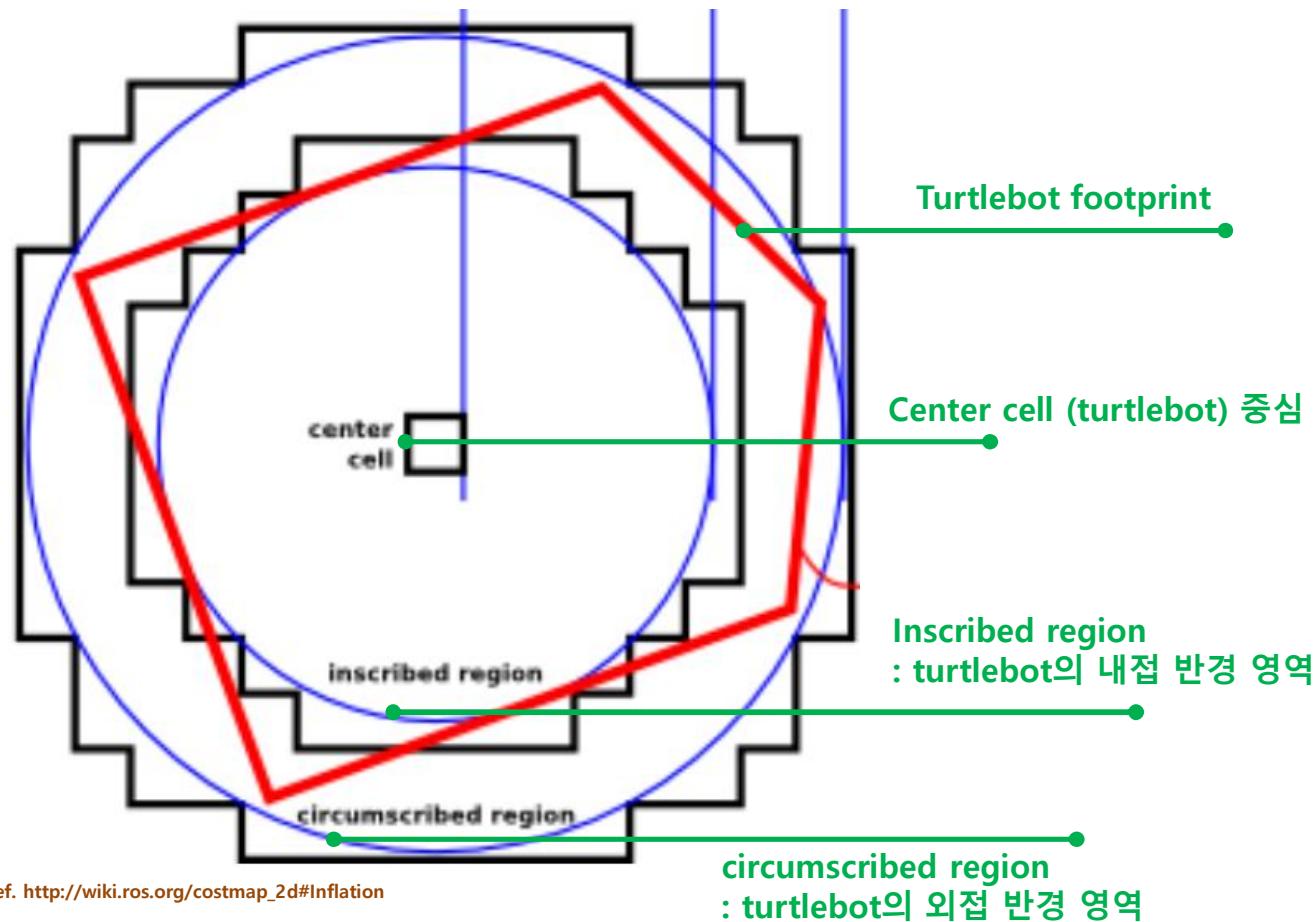


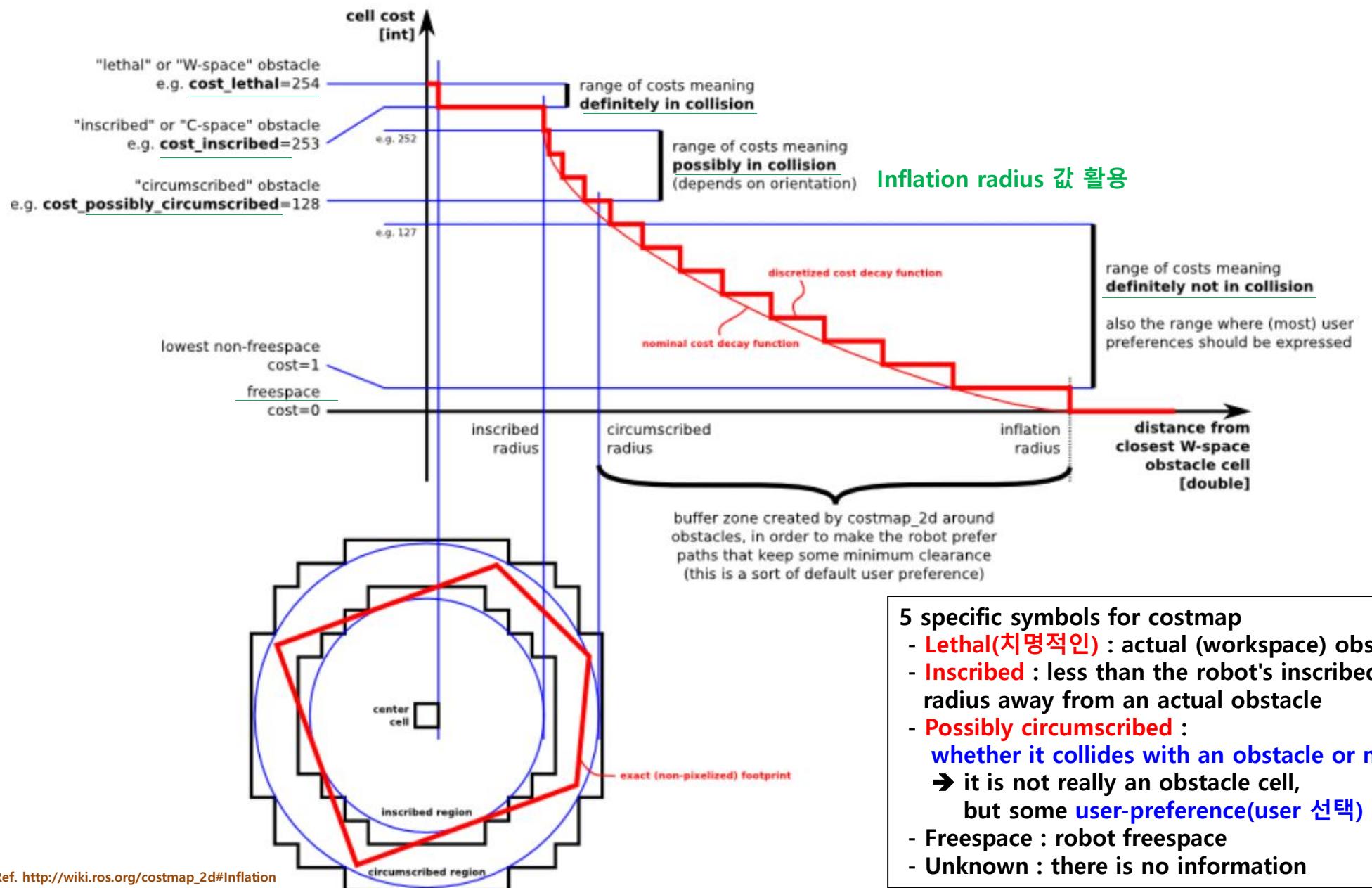
Navigation 실행 Rviz 화면



7-6. Costmap

(4) 장애물과의 거리와 costmap 값의 관계도





7-6. Costmap

(5) costmap 설정 parameters

: 장애물 영역, 장애물과 충돌이 예상되는 영역, robot이 이동 가능한 영역 계산 parameters

: http://wiki.ros.org/dwa_local_planner

→ `costmap_common_params_burger.yaml` : 공통 parameters

→ `global_costmap_params.yaml` : 전역 영역 모션 계획에 필요한 parameters

→ `local_costmap_params.yaml` : 국부 영역 모션 계획에 필요한 parameters

Name	Size	Type
<code>base_local_planner_params.yaml</code>	483 bytes	YAML document
<code>costmap_common_params_burger.yaml</code>	333 bytes	YAML document
<code>costmap_common_params_waffle.yaml</code>	332 bytes	YAML document
<code>costmap_common_params_waffle_pi.yaml</code>	332 bytes	YAML document
<code>dwa_local_planner_params_burger.yaml</code>	912 bytes	YAML document
<code>dwa_local_planner_params_waffle.yaml</code>	911 bytes	YAML document
<code>dwa_local_planner_params_waffle_pi.yaml</code>	911 bytes	YAML document
<code>global_costmap_params.yaml</code>	172 bytes	YAML document
<code>local_costmap_params.yaml</code>	243 bytes	YAML document
<code>move_base_params.yaml</code>	205 bytes	YAML document

opt	ros	kinetic	share	turtlebot3_navigation	param
					<code>base_local_planner_params.yaml</code> <code>costmap_common_params_burger.yaml</code> <code>costmap_common_params_waffle.yaml</code> <code>costmap_common_params_waffle_pi.yaml</code> <code>dwa_local_planner_params_burger.yaml</code> <code>dwa_local_planner_params_waffle.yaml</code> <code>dwa_local_planner_params_waffle_pi.yaml</code> <code>global_costmap_params.yaml</code> <code>local_costmap_params.yaml</code> <code>move_base_params.yaml</code>

(5) costmap 설정 parameters

① costmap_common_params_burger.yaml

Navigation에 설정된 map resolution에 따라

Navigation 실행중(Rviz에서), map 업데이트가 너무 느린 경우 발생

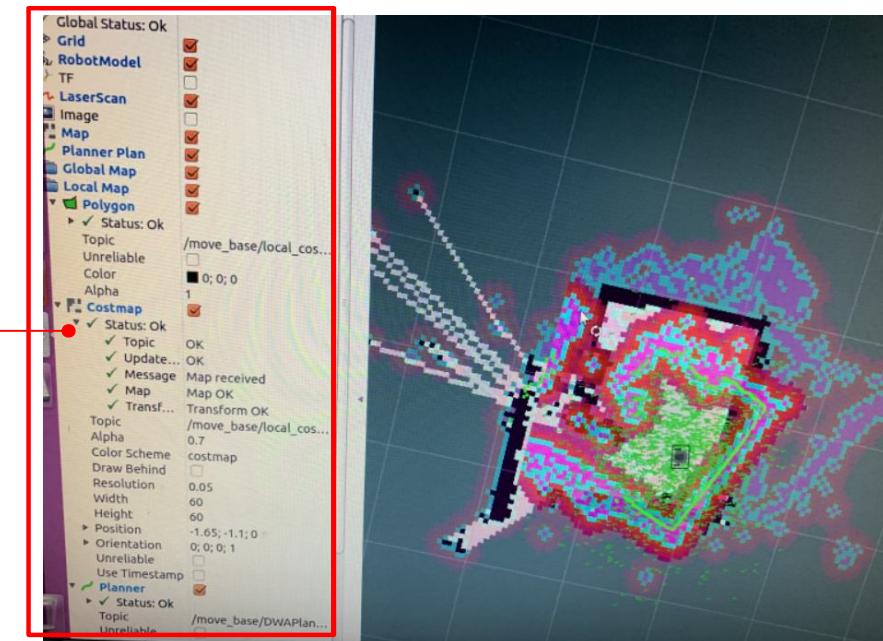
→ PC 리소스가 부족하여 발생함

→ obstacle_range 과 raytrace_range 값을 적절하게 낮추어야 함

→ Map의 크기가 PC의 리소스 한계를 넘어섬 (MAP 줄이기)

단위 [m]

```
costmap_common_params_burger.yaml 333 bytes YAML document 11/03/2021
costmap_common_params_burger.yaml ~/catkin_ws/src/turtlebot3/turtlebot3_navigation/param
File Edit View Search Tools Documents Help
obstacle_range: 3.0
raytrace_range: 3.5
footprint: [[-0.105, -0.105], [-0.105, 0.105], [0.041, 0.105], [0.041, -0.105]]
#robot_radius: 0.105
inflation_radius: 0.2
cost_scaling_factor: 3.0
map_type: costmap
observation_sources: scan
scan: {sensor_frame: base_scan, data_type: LaserScan, topic: scan,
marking: true, clearing: true}
```



object가 robot과 적용된 거리 값 내에 있으면 obstacle(장애물)로 처리

센서값 중에서
거리 이상의 데이터는 자유공간(freespace)으로 처리

robot의 외형 치수를 여러 개의 점인 polygon 형태

obstacle에 접근 못하게 하는 영역(obstacle 충돌 영역)

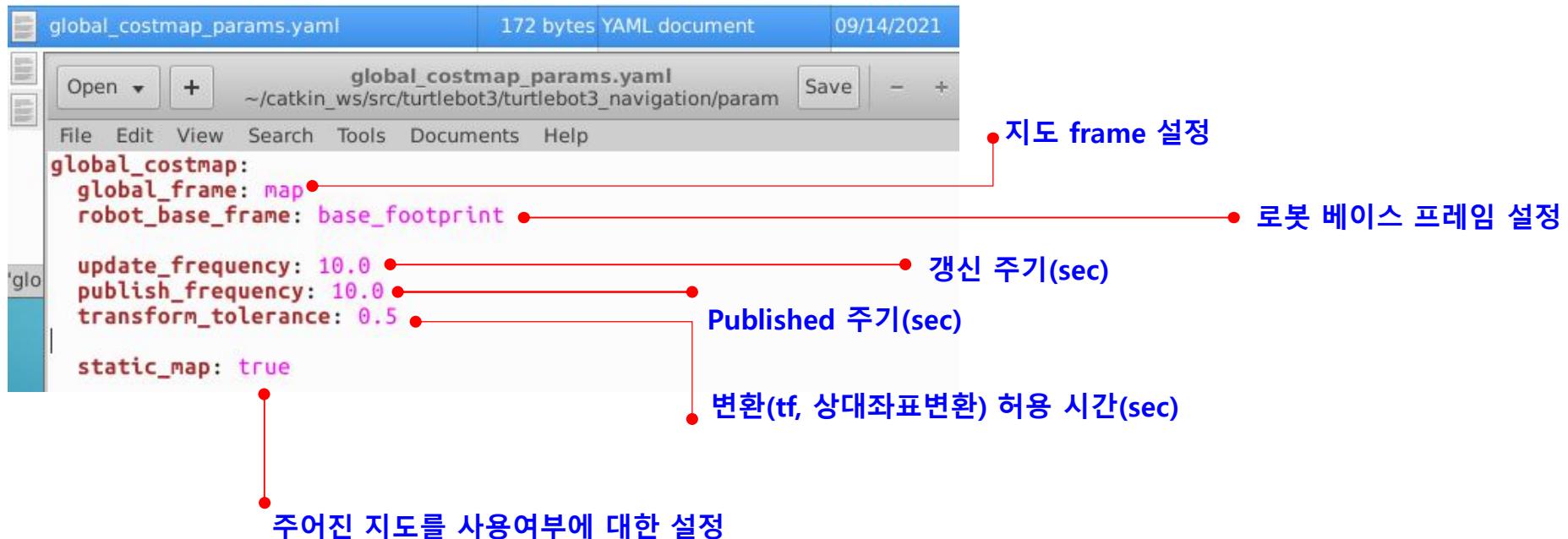
Laser scan의 data형, topic 형태, costmap에 반영, 최소 장애물 높이 설정

7. Navigation & SLAM (Simultaneous Localization and Mapping)

(5) costmap 설정 parameters

② `global_costmap_params.yaml` : 전역 영역 모션 계획에 필요한 parameters

단위 [m]

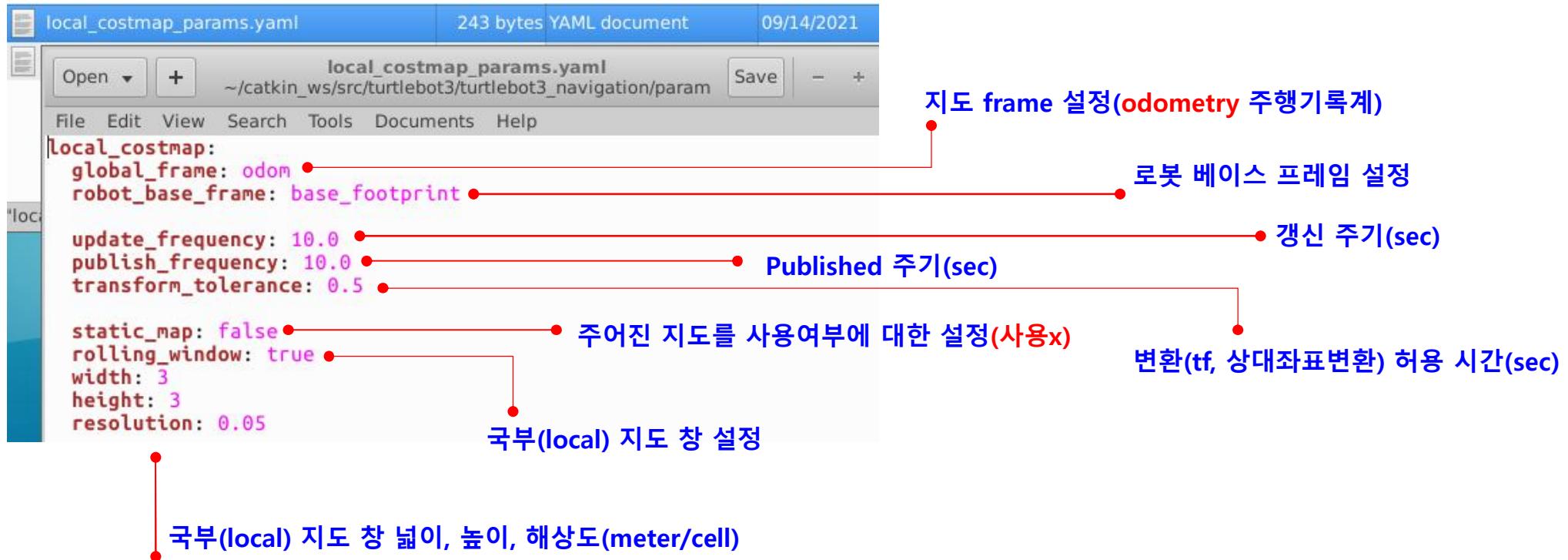


7. Navigation & SLAM (Simultaneous Localization and Mapping)

(5) costmap 설정 parameters

③ local_costmap_params.yaml : 국부 영역 모션 계획에 필요한 parameters

단위 [m]



(6) 그 외 설정 parameters

① move_base_params.yaml

: motion planning(path planning)을 총괄하는 move_base의 parameters 설정

[move_base 역할]

: path planning에 필요한 costmap parameters

: 이동속도 명령을 robot에 넘기는 dwa_local_planner에 대한 parameters

```

move_base_params.yaml
205 bytes YAML document 09/14/2021
move_base_params.yaml
~/catkin_ws/src/turtlebot3/turtlebot3_navigation/param
File Edit View Search Tools Documents Help
shutdown_costmaps: false
controller_frequency: 10.0
planner_patience: 5.0
controller_patience: 15.0
conservative_reset_dist: 3.0
planner_frequency: 5.0
oscillation_timeout: 10.0
oscillation_distance: 0.2

```

위 거리를 움직이면
oscillation_timeout 초기화 됨 [단위 m]

복구 동작(recovery behavior)을 위해
로봇이 왔다 갔다 하는 것을 허용하는 시간
[단위 sec]

전역 계획의 반복 주기[단위Hz]

/home/yongseok/catkin_ws/src/turtlebot3/turtlebot3_navigation/param/		
Name	Size	Type
base_local_planner_params.yaml	483 bytes	YAML document
costmap_common_params_burger.yaml	333 bytes	YAML document
costmap_common_params_waffle.yaml	332 bytes	YAML document
costmap_common_params_waffle_pi.yaml	332 bytes	YAML document
dwa_local_planner_params_burger.yaml	912 bytes	YAML document
dwa_local_planner_params_waffle.yaml	911 bytes	YAML document
dwa_local_planner_params_waffle_pi.yaml	911 bytes	YAML document
global_costmap_params.yaml	172 bytes	YAML document
local_costmap_params.yaml	243 bytes	YAML document
move_base_params.yaml	205 bytes	YAML document

move_base가 비활성화 상태일 때
costmap node를 정지시킬 것인가에 대한 여부 선택

Robot base에 속도 명령을 주는
Controller의 반복 주기[단위Hz]

global_costmap
(전역 이동 경로 계획) 반복 주기[단위Hz]

space-clearing 수행전
Controller의 최대 대기시간[단위 sec]

costmap 초기화 때
이 거리보다 먼 장애물은 지도에서 삭제 [단위 m]

(6) 그 외 설정 parameters

② base_local_params.yaml

```
base_local_planner_params.yaml 483 bytes YAML document
base_local_planner_params...
~/catkin_ws/src/turtlebot3/turt...
File Edit View Search Tools Documents Help
TrajectoryPlannerROS:
# Robot Configuration Parameters
max_vel_x: 0.18
min_vel_x: 0.08

max_vel_theta: 1.0
min_vel_theta: -1.0
min_in_place_vel_theta: 1.0

acc_lim_x: 1.0
acc_lim_y: 0.0
acc_lim_theta: 0.6

# Goal Tolerance Parameters
xy_goal_tolerance: 0.10
yaw_goal_tolerance: 0.05

# Differential-drive robot configuration
holonomic_robot: false

# Forward Simulation Parameters
sim_time: 0.8
vx_samples: 18
vtheta_samples: 20
sim_granularity: 0.05
```

→ 터틀봇3에서 dwa_local_planner_paramsburger.yaml을 적용

Name	Size	Type
base_local_planner_params.yaml	483 bytes	YAML document
costmap_common_params_burger.yaml	333 bytes	YAML document
costmap_common_params_waffle.yaml	332 bytes	YAML document
costmap_common_params_waffle_pi.yaml	332 bytes	YAML document
dwa_local_planner_params_burger.yaml	912 bytes	YAML document
dwa_local_planner_params_waffle.yaml	911 bytes	YAML document
dwa_local_planner_params_waffle_pi.yaml	911 bytes	YAML document
global_costmap_params.yaml	172 bytes	YAML document
local_costmap_params.yaml	243 bytes	YAML document
move_base_params.yaml	205 bytes	YAML document

7. Navigation & SLAM (Simultaneous Localization and Mapping)

7-7. Dynamic Window Approach

(1) definition

: 이동 경로 계획 및 장애물을 회피할 때 사용되는 방법

: Robot 속도 탐색 영역(velocity search space)에서
로봇과 충돌 가능한 장애물을 회피하면서 목표점까지 빠르게 다다를 수 있는 속도를 선택하는 방법

→ Dynamic Window Approach 라고 함

: 공간에서 로봇은 하드웨어 한계로 최대 허용 속도가 존재하고 이를 Dynamic Window 라고 함

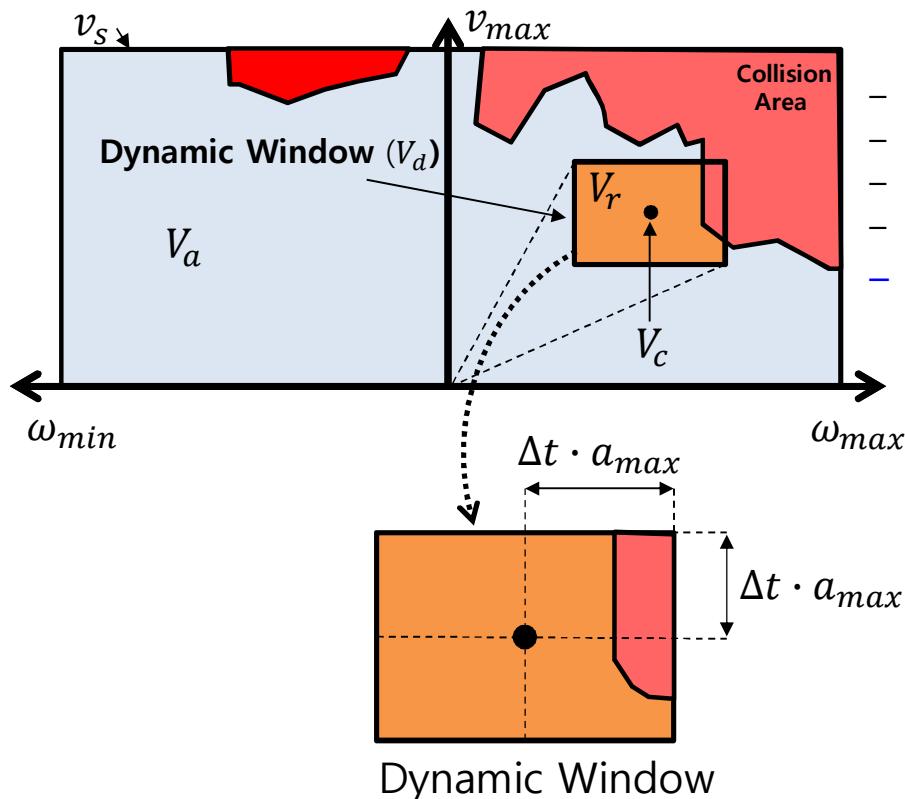
: ROS에서 지역 이동 계획에 Trajectory(궤도) planner를 많이 사용했으나 최근 DWA가 성능에서 뛰어나 대체되고 있음

→ dwa_local_planner_params_burger.yaml : 최종적으로 이동속도 명령을 로봇에 넘기는 패키지

7-7. Dynamic Window Approach

(2) 속도 탐색 영역(velocity search space) 과 Dynamic Window

: Robot x, y축 위치 좌표가 아니라 v (병진 속도), ω (회전 속도)를 축으로 하는 속도 탐색 영역(velocity search space)에서 Robot은 하드웨어 한계로 최대 허용 속도가 존재하고 이를 Dynamic Window 라고 함



- v (병진속도), ω (회전속도)
- V_s : 가능 속도 영역
- V_a : 허용 속도 영역
- V_r : Dynamic window 안의 속도 영역 V_c : 현재속도
- $G_{(v,\omega)} = \sigma(\alpha \cdot heading(v, \omega) + \beta \cdot dist(v, \omega) + \gamma \cdot velocity(v, \omega))$

→ 목적 함수 G

: 목적 함수 G는 로봇의 방향, 속도, 충돌을 고려하여,
목적 함수가 최대가 되는 v (병진속도), ω (회전속도)를 구함

- a_{max} : 최대 가감 속도
- $heading(v, \omega)$: $180 - (\text{로봇의 방향과 목표 지점의 방향 차})$
- $dist(v, \omega)$: 장애물과의 거리
- $velocity(v, \omega)$: 선택한 속도
- σ, β, γ : 가중치 상수

Ref. D. Fox, W. Burgard The dynamic window approach to collision avoidance, IEEE Robotics & Automation Magazine

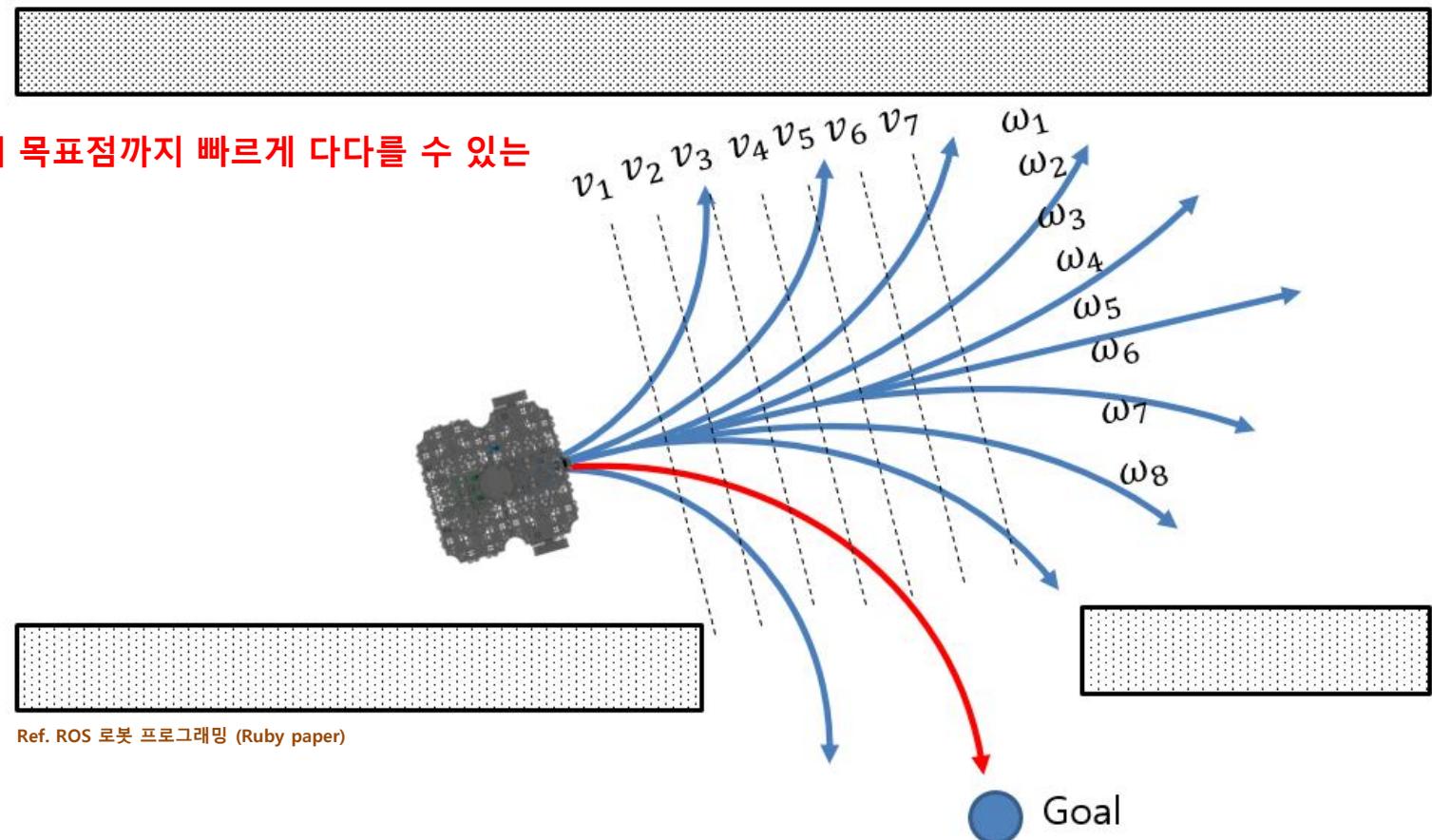
7. Navigation & SLAM (Simultaneous Localization and Mapping)

7-7. Dynamic Window Approach

(2) 속도 탐색 영역(velocity search space) 과 Dynamic Window

: v (병진 속도), ω (회전 속도)

충돌 가능한 장애물을 회피하면서 목표점까지 빠르게 다다를 수 있는 속도를 선택하는 방법



(3) 설정 parameters

dwa_local_planner_params_burger.yaml

: 최종적으로 이동속도 명령을 로봇에 넘기는 패키지

최대, 최소 회전 속도(meter/sec)

x, y 가속도 제한(meter/sec²)

Theta축 각가속도 제한(radian/sec²)

x, y 거리 목표지점 허용오차(meter)

yaw축 목표지점 허용오차(radian)

Latch_xy_goal_tolerance : false

If goal tolerance is latched(true),
if the robot ever reaches the goal xy location it will
simply rotate in place,
even if it ends up outside the goal tolerance



```
DWAPlannerROS:
# Robot Configuration Parameters
max_vel_x: 0.22
min_vel_x: -0.22
max_vel_y: 0.0
min_vel_y: 0.0

# The velocity when robot is moving in a straight line
max_trans_vel: 0.22
min_trans_vel: 0.11
max_rot_vel: 2.75
min_rot_vel: 1.37
acc_lim_x: 2.5
acc_lim_y: 0.0
acc_lim_theta: 3.2

# Goal Tolerance Parameters
xy_goal_tolerance: 0.05
yaw_goal_tolerance: 0.17
latch_xy_goal_tolerance: false

# Forward Simulation Parameters
sim_time: 1.5
vx_samples: 20
vy_samples: 0
vth_samples: 40
controller_frequency: 10.0

# Trajectory Scoring Parameters
path_distance_bias: 32.0
goal_distance_bias: 20.0
occdist_scale: 0.02
forward_point_distance: 0.325
stop_time_buffer: 0.2
scaling_speed: 0.25
max_scaling_factor: 0.2

# Oscillation Prevention Parameters
oscillation_reset_dist: 0.05

# Debugging
publish_traj_pc : true
publish_cost_grid_pc: true
```

x축 최대, 최소 속도(meter/sec)

전방향(omni-direction)
로봇 경우에 적용



Mecanum wheel

dwa_local_planner_params_burger.yaml

: 최종적으로 이동속도 명령을 로봇에 넘기는 패키지

전방향 시뮬레이션 궤적 시간
x 축 속도 공간에서 탐색하는 sample 수
y 축 속도 공간에서 탐색하는 sample 수
theta 축 속도 공간에서 탐색하는 sample 수

컨트롤러가 주어진 경로를 얼마나 따르는 가에 대한 가중치
목표지점과 제어 속도에 근접한 지에 대한 가중치
장애물 회피에 대한 가중치
로봇 중점과 추가 scoring point와의 거리(meter)
로봇이 충돌 전 정지에 필요한 시간(sec)
스케일링 속도(meter/sec)
최대 scaling 요소

궤적(Trajectory) Scoring parameters(궤적 평가)
궤적 평가를 위한 비용 함수에 사용될 스코어 계산

$\text{cost} = \text{path_distance_bias} * (\text{distance to path from the endpoint of the trajectory in meters})$
+ $\text{goal_distance_bias} * (\text{distance to local goal from the endpoint of the trajectory in meters})$
+ $\text{occdist_scale} * (\text{maximum obstacle cost along the trajectory in obstacle cost (0-254)})$

```
# Forward Simulation Parameters
sim_time: 1.5
vx_samples: 20
vy_samples: 8
vth_samples: 40
controller_frequency: 10.0

# Trajectory Scoring Parameters
path_distance_bias: 32.0
goal_distance_bias: 20.0
occdist_scale: 0.02
forward_point_distance: 0.325
stop_time_buffer: 0.2
scaling_speed: 0.25
max_scaling_factor: 0.2

# Oscillation Prevention Parameters
oscillation_reset_dist: 0.05

# Debugging
publish_traj_pc : true
publish_cost_grid_pc: true
```

이동 궤적 디버깅 설정
costmap 디버깅 설정

우왕자왕(Oscillation) 동작 방지 parameter
: Oscillation 플래그가 reset 전에 로봇 이동해야만 하는 거리

7. Navigation & SLAM (Simultaneous Localization and Mapping)

7-8. AMCL(Adaptive Monte Carlo Localization)

※ Particle Filter(SLAM 적용)

- 물체 추정에 최근 가장 많이 사용되는 알고리즘
 - : Noise가 있는 환경에서 측정된 data를 filter로 사용해 실제 위치를 추정하는 방법
- 시행착오(try and error)법에 기반한 simulation을 통한 예측 기술
 - : 여러가지 방법을 실험해, 결과들의 인과관계를 추론하여 문제를 해결하는 방식
- 대상 시스템에 확률 분포로, 임의 생성된 추정 값을 particle(입자) 형태로 나타내어 Particle Filter 라고 함
 - ➔ Monte Carlo Localization 방법이라 함

※ definition

: Monte Carlo Localization(위치 추정) 알고리즘에서,
적은 수의 샘플 수를 사용하여 수행 시간을 줄여 실시간성을 높인 위치 추정 알고리즘

- ➔ AMCL(Adaptive Monte Carlo Localization)
 - : 상세 내용 Sebastian Thrun 'Probabilistic Robotics' 저서

7-9. Navigation : turtlebot3_navigation.launch

The screenshot shows a file browser window with the path `/home/yongseok/catkin_ws/src/turtlebot3/turtlebot3_navigation/launch/`. The file `turtlebot3_navigation.launch` is open in a text editor. The code defines a launch configuration with several arguments and includes files for AMCL, move_base, and rviz.

```
<launch>
  <!-- Arguments -->
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
  <arg name="map_file" default="$(find turtlebot3_navigation)/maps/map.yaml"/>
  <arg name="open_rviz" default="true"/>
  <arg name="move_forward_only" default="false"/>

  <!-- Turtlebot3 -->
  <include file="$(find turtlebot3_bringup)/launch/turtlebot3_remote.launch">
    <arg name="model" value="$(arg model)" />
  </include>

  <!-- Map server -->
  <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)"/>

  <!-- AMCL -->
  <include file="$(find turtlebot3_navigation)/launch/amcl.launch"/>

  <!-- move_base -->
  <include file="$(find turtlebot3_navigation)/launch/move_base.launch">
    <arg name="model" value="$(arg model)" />
    <arg name="move_forward_only" value="$(arg move_forward_only)"/>
  </include>

  <!-- rviz -->
  <group if="$(arg open_rviz)">
    <node pkg="rviz" type="rviz" name="rviz" required="true"
          args="-d $(find turtlebot3_navigation)/rviz/turtlebot3_navigation.rviz"/>
  </group>
</launch>
```

AMCL
(Adaptive Monte Carlo Localization)

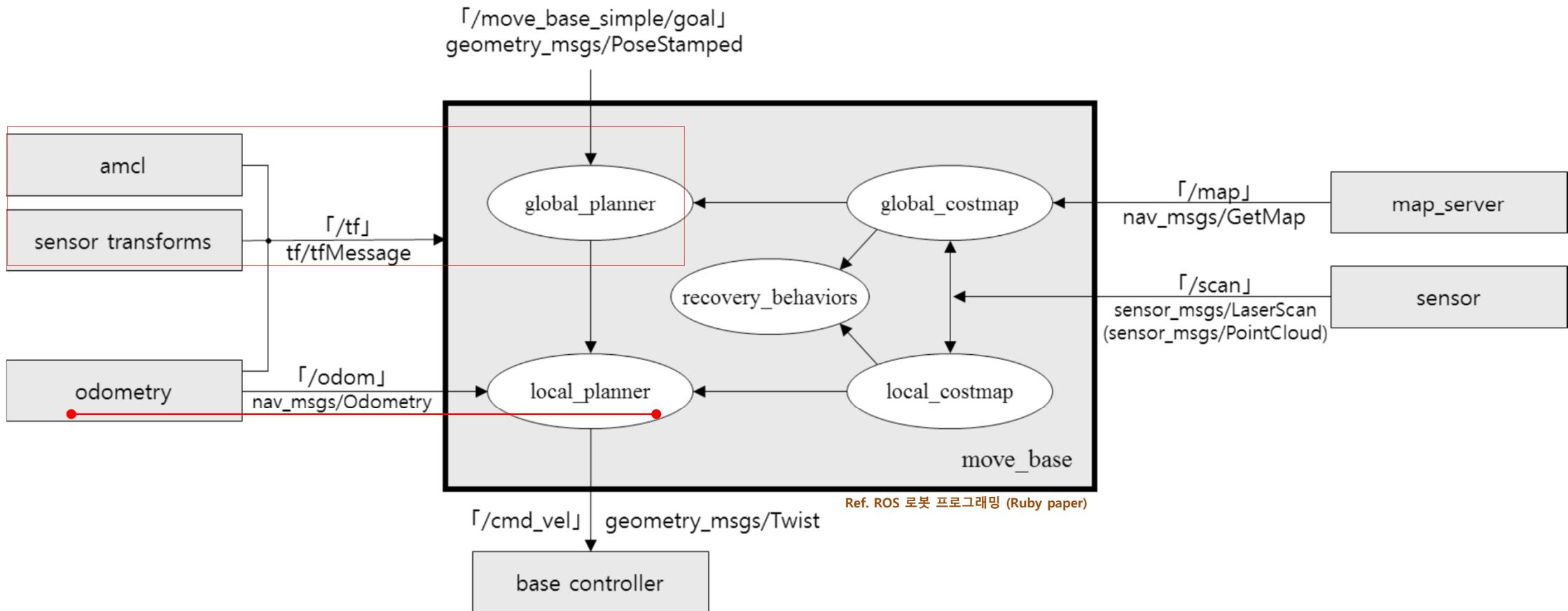
move_base

: path planning에 필요한 costmap parameters

: 이동속도 명령을 robot에 넘기는 dwa_local_planner에 대한 parameters

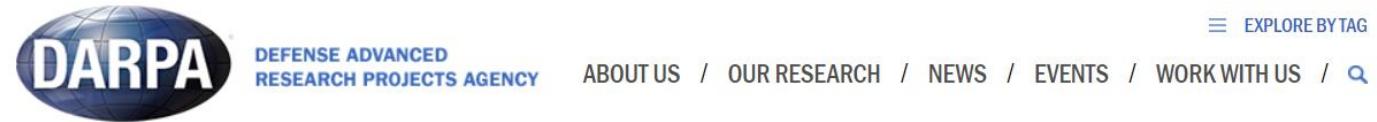
: path planning을 총괄하는 move_base의 parameters를 제어

7-10. Navigation stack 설정에 관한 각 필수 노드와 토픽 관계도



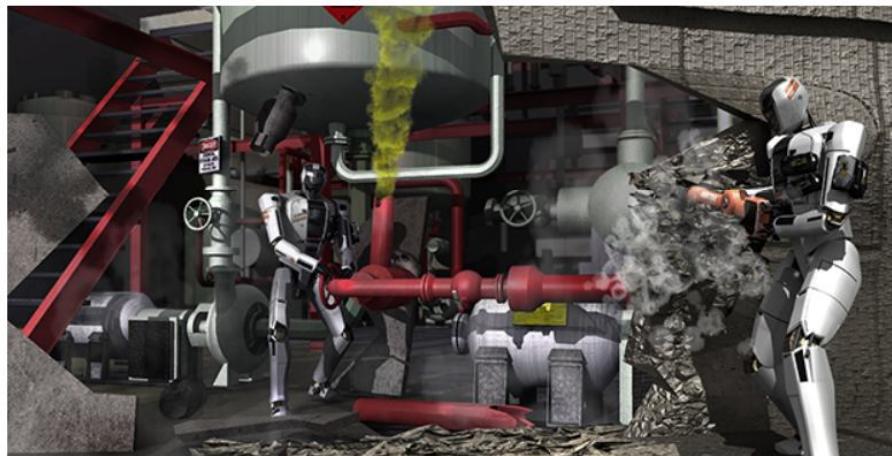
7-11. Gazebo Simulation

: 미국 DARPA Robotics Challenge (<https://www.darpa.mil/program/darpa-robotics-challenge>)



> Defense Advanced Research Projects Agency > Our Research > DARPA Robotics Challenge

DARPA Robotics Challenge (DRC) (Archived)



[제목 없음]

The Department of Defense's strategic plan calls for the Joint Force to conduct humanitarian, disaster relief, and related operations. Some disasters, due to grave risks to the health and wellbeing of rescue and aid workers, prove too great in scale or scope for timely and effective human response. The DARPA Robotics Challenge (DRC) seeks to address this problem by promoting innovation in human-supervised robotic technology for disaster-response operations.

The primary technical goal of the DRC is to develop human-supervised ground robots capable of executing complex tasks in dangerous, degraded, human-engineered environments. Competitors in the DRC are developing robots that can utilize standard tools and equipment commonly available in human environments, ranging from hand tools to vehicles.

7-11. Gazebo Simulation

(1) definition

: 미국 DARPA Robotics Challenge 공식(<https://www.darpa.mil/program/darpa-robotics-challenge>) 시뮬레이터

- 로봇공학분야에서 활용이 높은 simulator

- ROS와 Gazebo의 호환성 탁월

- Gazebo는 3차원 시뮬레이터로 물리 엔진과 그래픽 효과 사용에 따른 CPU 및 GPU, RAM 사용량 많음

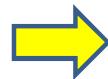
→ 사용 중인 PC의 사양에 따라 로딩에 상당한 시간 소요(remote PC 사양과 네트워크 용량에 따라) ★

새터미널에서

```
$ export TURTLEBOT3_MODEL=burger
```

```
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

실행 화면 (로딩 시간이 오래 걸림)



(2) 실습 : e-manual 6장 <https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/>

ⓐ TurtleBot3 Simulation Package 설치하기

새터미널에서

```
$ cd ~/catkin_ws/src/
```

```
$ git clone -b noetic-devel https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
```

```
yongseok@yongseok:~$ cd ~/catkin_ws/src/
yongseok@yongseok:~/catkin_ws/src$ git clone -b kinetic-devel
  https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
Cloning into 'turtlebot3_simulations'...
remote: Enumerating objects: 2959, done.
remote: Counting objects: 100% (572/572), done.
remote: Compressing objects: 100% (241/241), done.
Receiving objects: 15% (444/2959), 388.01 KiB | 679.00 KiB/s
Receiving objects: 16% (474/2959), 388.01 KiB | 679.00 KiB/s
Receiving objects: 17% (504/2959), 388.01 KiB | 679.00 KiB/s 생략
Receiving objects: 100% (2959/2959), 14.32 MiB | 6.93 MiB/s
Receiving objects: 100% (2959/2959), 15.36 MiB | 6.93 MiB/s,
done.
Resolving deltas: 100% (1685/1685), done.
Checking connectivity... done.
```

```
$ cd ~/catkin_ws && catkin make
```

```
checking connection...  
yongseok@yongseok:~/catkin_ws/src$ cd ~/catkin_ws && catkin_make  
Base path: /home/yongseok/catkin_ws  
Source space: /home/yongseok/catkin_ws/src  
Build space: /home/yongseok/catkin_ws/build  
Devel space: /home/yongseok/catkin_ws/devel  
Install space: /home/yongseok/catkin_ws/install  
####  
#### Running command: "cmake /home/yongseok/catkin_ws/src -DCATKIN_DEVEL_PREFIX=/home/yongseok/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/home/yongseok/catkin_ws/install"  
-- The C compiler identification is unknown  
-- The CXX compiler identification is unknown  
-- Check for working C compiler: /usr/bin/c  
-- Check for working C compiler: /usr/bin/c -- works  
-- Detecting C compiler ABI info  
-- Detecting C compiler ABI info - done  
-- The CXX compiler identification is unknown  
-- Check for working CXX compiler: /usr/bin/c++  
-- Check for working CXX compiler: /usr/bin/c++ -- works  
-- Detecting CXX compiler ABI info  
-- Detecting CXX compiler ABI info - done  
-- Configuring done  
-- Generating done  
-- Build files have been written to: /home/yongseok/catkin_ws/build
```

```
-- ~~ traversing 19 packages in topological order:
-- ~~ - turtlebot3 (metapackage)
-- ~~ - turtlebot3_applications (metapackage)
-- ~~ - turtlebot3_follow_filter
-- ~~ - turtlebot3_msgs
-- ~~ - turtlebot3_navigation
-- ~~ - turtlebot3_simulations (metapackage)

-- +++ processing catkin package: 'turtlebot3_gazebo'
-- ==> add subdirectory(turtlebot3_simulations/turtlebot3_gazebo)
-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy
-- Boost version: 1.58.0
-- Found the following Boost libraries:
--   thread
--   signals
--   system
```

생략

생략

```
-- Build files have been written to: /home/yongseok/catkin_ws  
/build  
####  
#### Running command: "make -j2 -l2" in "/home/yongseok/catki  
n_ws/build"  
####  
[ 0%] Built target std_msgs_generate_messages_py I  
[ 0%] Built target _turtlebot3_msgs_generate_messages_check_  
deps_SensorState
```

생략

```
sages
[ 98%] Built target turtlebot3_example_generate_messages
[100%] Linking CXX executable /home/yongseok/catkin_ws/devel/
lib/turtlebot3_fake/turtlebot3_fake_node
[100%] Built target turtlebot3_fake_node
yongseok@yongseok:~/catkin_ws$
```

(2) 실습 : e-manual 6장 <https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/>

⑥ Gazebo Simulation 실행하기(turtlebot3_empty_world)

새터미널에서

```
$ export TURTLEBOT3_MODEL=burger
```

```
$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
```

```
[yongseok@yongseok: ~]$ export TURTLEBOT3_MODEL=burger
[yongseok@yongseok: ~]$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
... logging to /home/yongseok/.ros/log/055f23ba-4783-11ec-alb2-f40669f0af37/roslaunch-yongseok-7319.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.0.30:32773/
=====

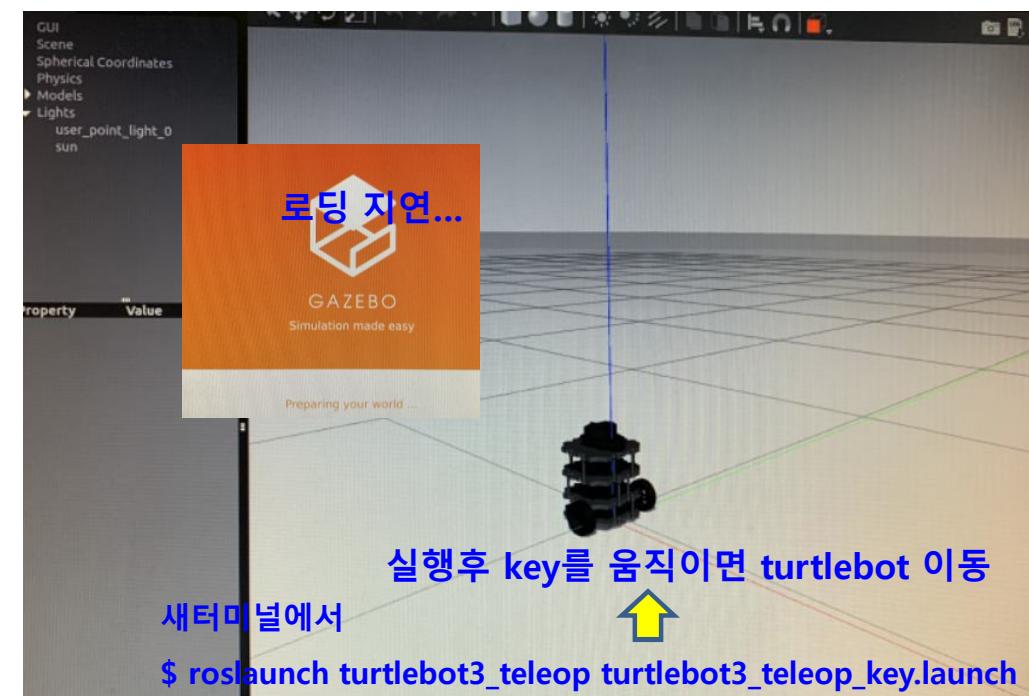
SUMMARY
=====

PARAMETERS
* /robot_description: <?xml version="1.....
* /readiness_level: 1.000000
* /use_sim_time: true
* /use_sim�
[yongseok@yongseok: ~]$ /home/yongseok/catkin_ws/src/turtlebot3_simulations/turtlebot3_gazebo
gazebo (gazebo_ros/gzserver)
gazebo_gui (gazebo_ros/gzclient)
spawn_urdf (gazebo_ros/spawn_model)

auto-starting new master
process[master]: started with pid [7332]
ROS_MASTER_URI=http://192.168.0.30:11311

setting /run_id to 055f23ba-4783-11ec-alb2-f40669f0af37
process[rosout-1]: started with pid [7345]
started core service [/rosout]
process[gazebo-2]: started with pid [7353]
process[gazebo_gui-3]: started with pid [7372]
process[spawn_urdf-4]: started with pid [7378]
[INFO] [1637138862.520510322]: Finished loading Gazebo ROS API plugin
```

생략



실행후 key를 움직이면 turtlebot 이동

새터미널에서

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

(2) 실습 : e-manual 6장 <https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/>

② Gazebo Simulation 실행하기(turtlebot3_world.launch)

앞에서 실행한 창을 모두 닫기 (CTL + C), Gazebo 닫기

새터미널에서

```
$ export TURTLEBOT3_MODEL=burger
```

```
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

```
yongseok@yongseok:~$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
... logging to /home/yongseok/.ros/log/7ae536c8-479d-11ec-a1b2-f40669f0af37
[roslaunch-yongseok-6402.log]
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.0.30:44283/
```

생략

```
auto-starting new master
process[master]: started with pid [6415]
ROS_MASTER_URI=http://192.168.0.30:11311

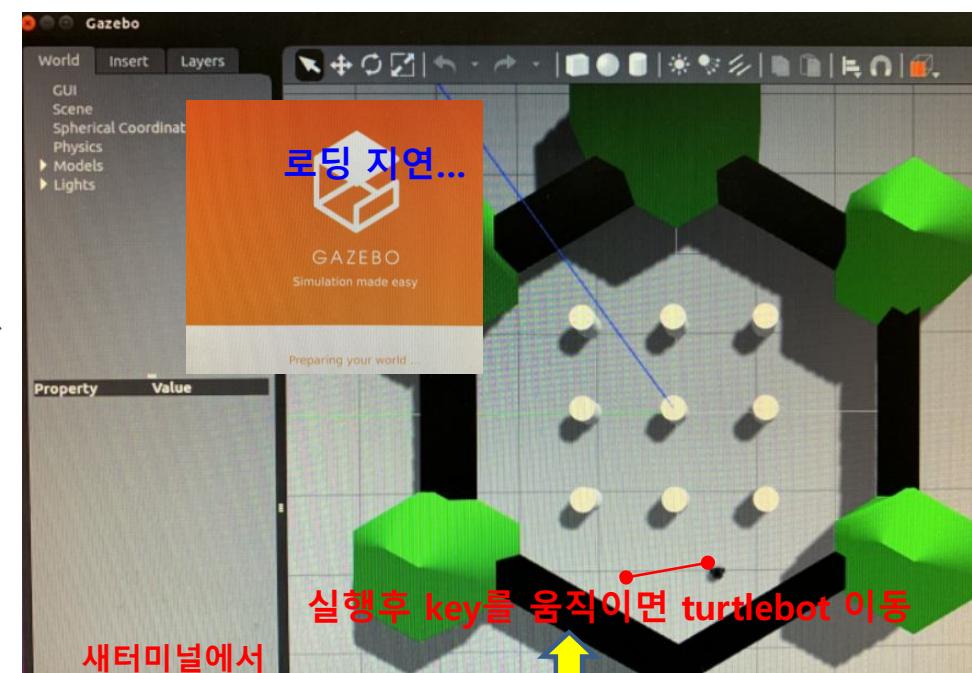
setting /run_id to 7ae536c8-479d-11ec-a1b2-f40669f0af37
process[rosout-1]: started with pid [6428]
started core service [/rosout]
process[gazebo-2]: started with pid [6435]
process[gazebo_gui-3]: started with pid [6450]
process[spawn_urdf-4]: started with pid [6459]
```

생략

```
[ INFO] [1637150228.74542698, 0.14600000]: Starting plugin DiffDrive(ns = //)
[ INFO] [1637150228.753252947, 0.14600000]: DiffDrive(ns = //): <rosDebugLevel> = na
[ INFO] [1637150228.758252990, 0.14600000]: DiffDrive(ns = //): <tf_prefix> =
[ INFO] [1637150228.760605032, 0.14600000]: DiffDrive(ns = //): Advertise joint_states
[ INFO] [1637150228.762137623, 0.14600000]: DiffDrive(ns = //): Try to subscribe to cmd_v
[ INFO] [1637150228.769410605, 0.14600000]: DiffDrive(ns = //): Subscribe to cmd_vel
[ INFO] [1637150228.770931149, 0.14600000]: DiffDrive(ns = //): Advertise odom on odom
[spawn_urdf-4] process has finished cleanly
log file: /home/yongseok/.ros/log/7ae536c8-479d-11ec-a1b2-f40669f0af37/spawn_urdf-4*.log
```

Model 명을 burger 또는 waffle

전망대 공간에 burger 또는 waffle 표시



새터미널에서

실행후 key를 움직이면 turtlebot 이동

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

(2) 실습 : e-manual 6장 <https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/>

④ Gazebo Simulation 실행하기(turtlebot3_house.launch)

앞에서 실행한 창을 모두 닫기 (CTL + C), Gazebo 닫기

새터미널에서

```
$ export TURTLEBOT3_MODEL=waffle_pi  
$ roslaunch turtlebot3_gazebo turtlebot3_house.launch
```

```
yongseok@yongseok:~$ roslaunch turtlebot3_gazebo turtlebot3_house.launch  
... logging to /home/yongseok/.ros/log/052f07c4-47a7-11ec-a1b2-f40669f0af37/  
/roslaunch-yongseok-8176.log  
Checking log directory for disk usage. This may take awhile.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
  
started roslaunch server http://192.168.0.30:44179/  
  
ROS_MASTER_URI=http://192.168.0.30:11311  
  
setting /run_id to 052f07c4-47a7-11ec-a1b2-f40669f0af37  
process[rosout-1]: started with pid [8202]  
started core service [/rosout]  
process[gazebo-2]: started with pid [8214]  
process[gazebo_gui-3]: started with pid [8226]  
process[spawn_urdf-4]: started with pid [8236]  
[ INFO] [1637154324.470026957]: Finished loading Gazebo ROS API Plugin  
[ INFO] [1637154513.788090247, 0.001000000]: DiffDrive(ns = //): Advertise jo  
[ INFO] [1637154513.790216925, 0.001000000]: DiffDrive(ns = //): Try to subsc  
[ INFO] [1637154513.806967426, 0.001000000]: DiffDrive(ns = //): Subscribe to  
[ INFO] [1637154513.810485463, 0.001000000]: DiffDrive(ns = //): Advertise od  
[spawn_urdf-4] process has finished cleanly  
log file: /home/yongseok/.ros/log/052f07c4-47a7-11ec-a1b2-f40669f0af37/spawn_
```

Turtlebot3 Waffle pi

집 공간에 Waffle pi 표시



실행후 key를 움직이면 turtlebot 이동

새터미널에서

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

7. Navigation & SLAM (Simultaneous Localization and Mapping)

7-11. Gazebo Simulation

(3) SLAM Simulation : e-manual 6장 https://emanual.robotis.com/docs/en/platform/turtlebot3/slam_simulation/

ⓐ Gazebo Simulation 실행하기(turtlebot3_world.launch)

앞에서 실행한 창을 모두 닫기 (CTL + C), Gazebo 닫기

새터미널에서

```
$ export TURTLEBOT3_MODEL=burger  
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```



Gazebo가 완전히(로딩 완료되면) 열린 후, 새터미널에서

```
$ export TURTLEBOT3_MODEL=burger  
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
```

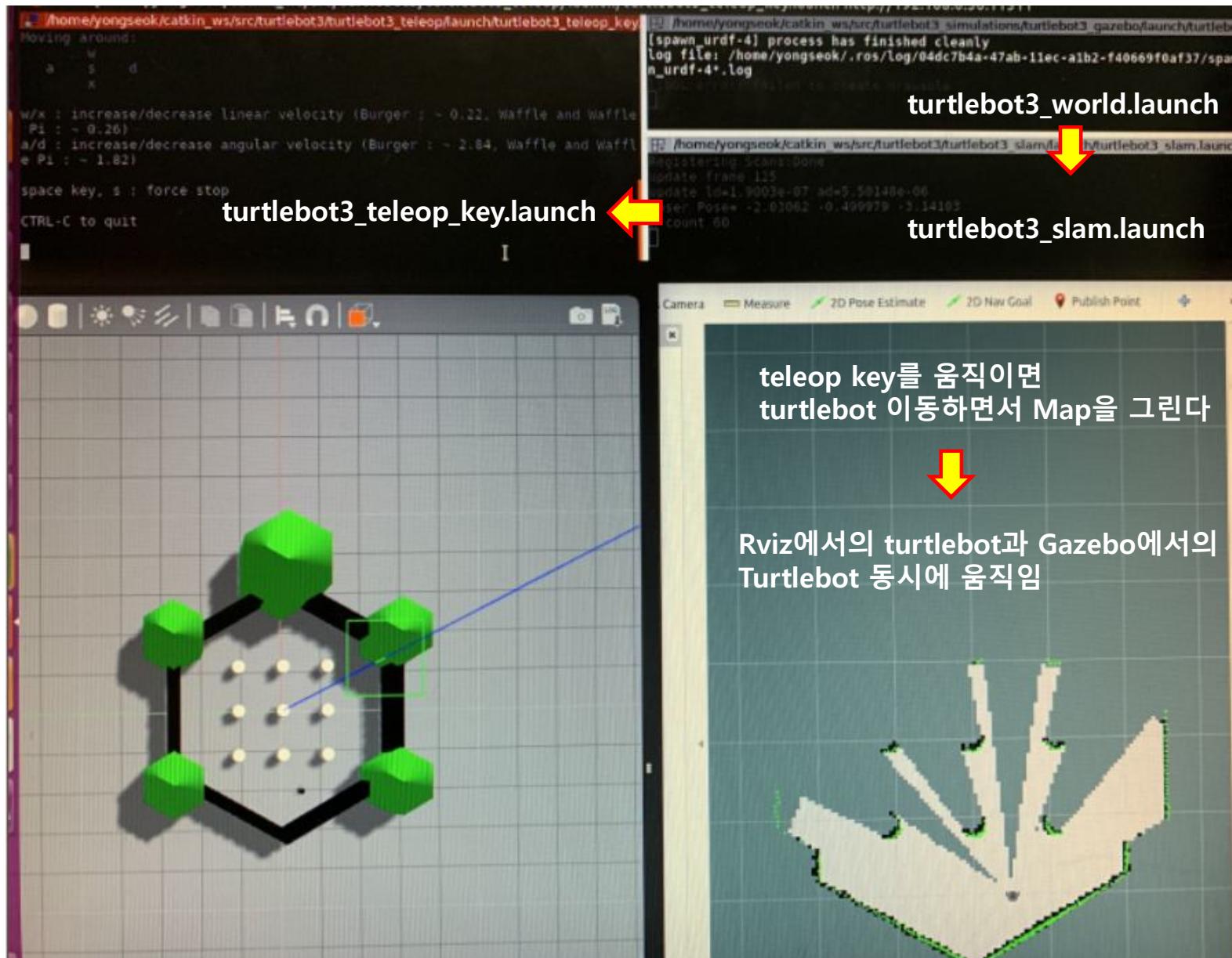


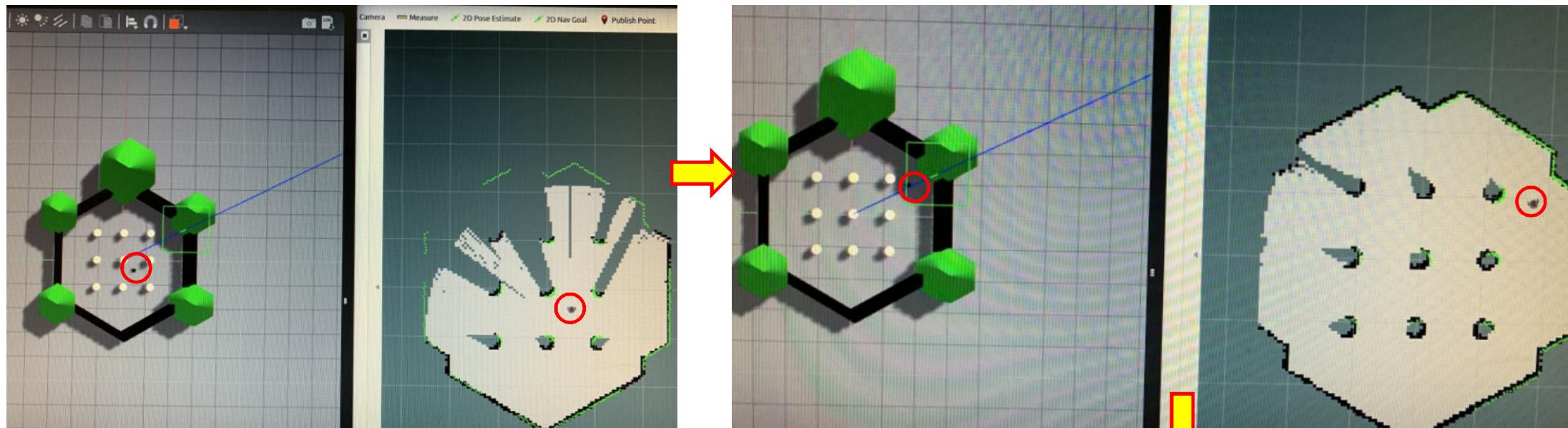
Rviz가 완전히 열린 후, 새터미널에서

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

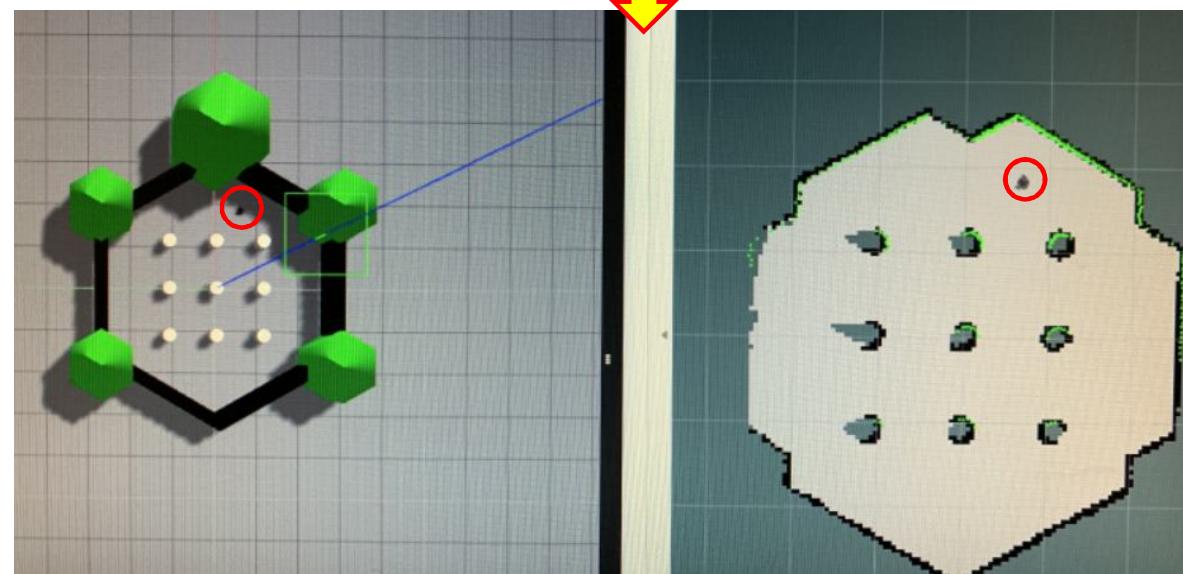


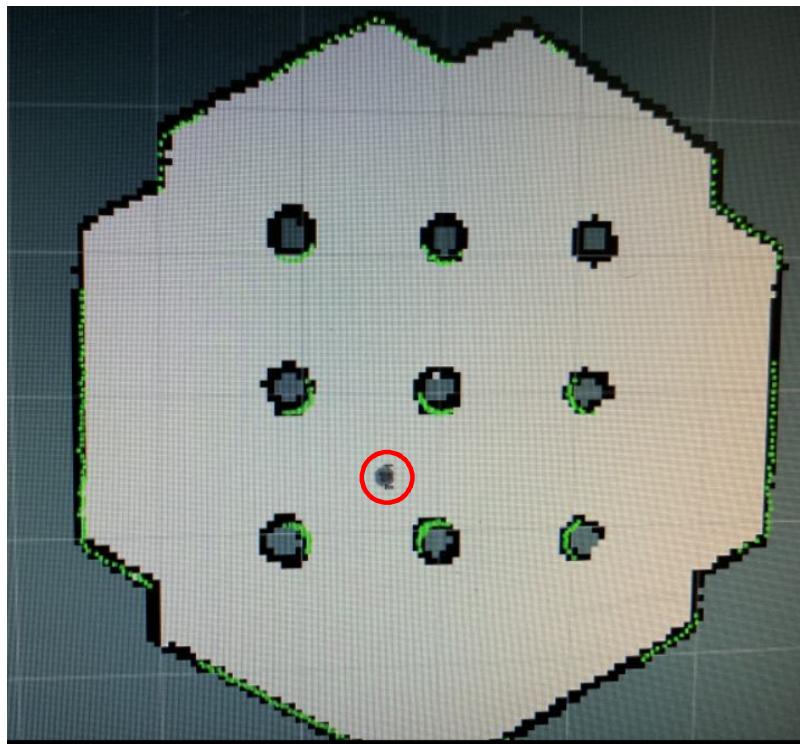
실행후 key를 움직이면 turtlebot 이동하면서 Map을 그린다





실행후 key를 움직이면 turtlebot 이동하면서
Map을 그린다





```
urrently: linear vel 0.14 angular vel 0.0
urrently: linear vel 0.15 angular vel 0.0
urrently: linear vel 0.16 angular vel 0.0
urrently: linear vel 0.17 angular vel 0.0
urrently: linear vel 0.18 angular vel 0.0
urrently: linear vel 0.19 angular vel 0.0
urrently: linear vel 0.2 angular vel 0.0
urrently: linear vel 0.21 angular vel 0.0
```

병진속도 증가 값

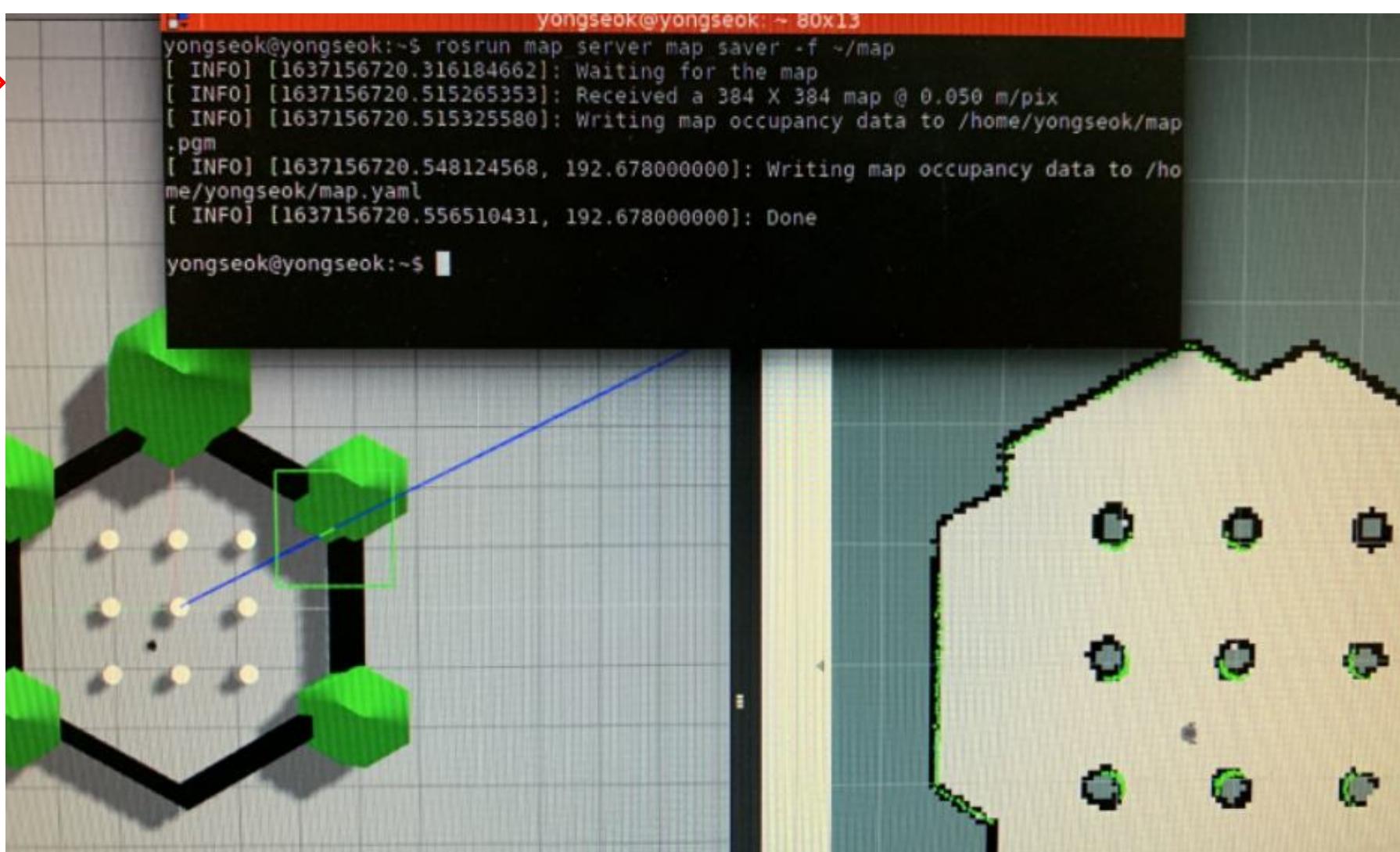
```
Registering Scans:Done
update frame 169
update ld=0.13212 ad=5.52266e-05
Laser Pose= 1.72713 -0.845051 -2.04526
m_count 103
```

Robot pose 값

새터미널에서

\$ rosrun map_server map_saver -f ~/map

Map 저장



7. Navigation & SLAM (Simultaneous Localization and Mapping)

7-11. Gazebo Simulation

(4) Navigation Simulation : e-manual 6장 https://emanual.robotis.com/docs/en/platform/turtlebot3/nav_simulation/

앞에서 실행한 창을 모두 닫기 (CTL + C), Gazebo 닫기, Rviz 닫기

새터미널에서

```
$ export TURTLEBOT3_MODEL=burger  
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```



Gazebo가 완전히 열린 후, 새터미널에서

```
$ export TURTLEBOT3_MODEL=burger  
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```



Rviz가 완전히 열린 후, 새터미널에서

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```



실행후 key를 움직이면 turtlebot 제자리 회전하면서 Map과 장애물을 일치 시킨다
또는 2D Pose Estimate를 활용하여 Map과 장애물을 일치 시킨다

```
roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```

실행후 Rviz 최초 화면

: Map과 장애물이 일치하지 않음

→ 2D Pose Estimate 이용하여 일치시키기

→ turtlebot3_teleop_key 를 활용하여

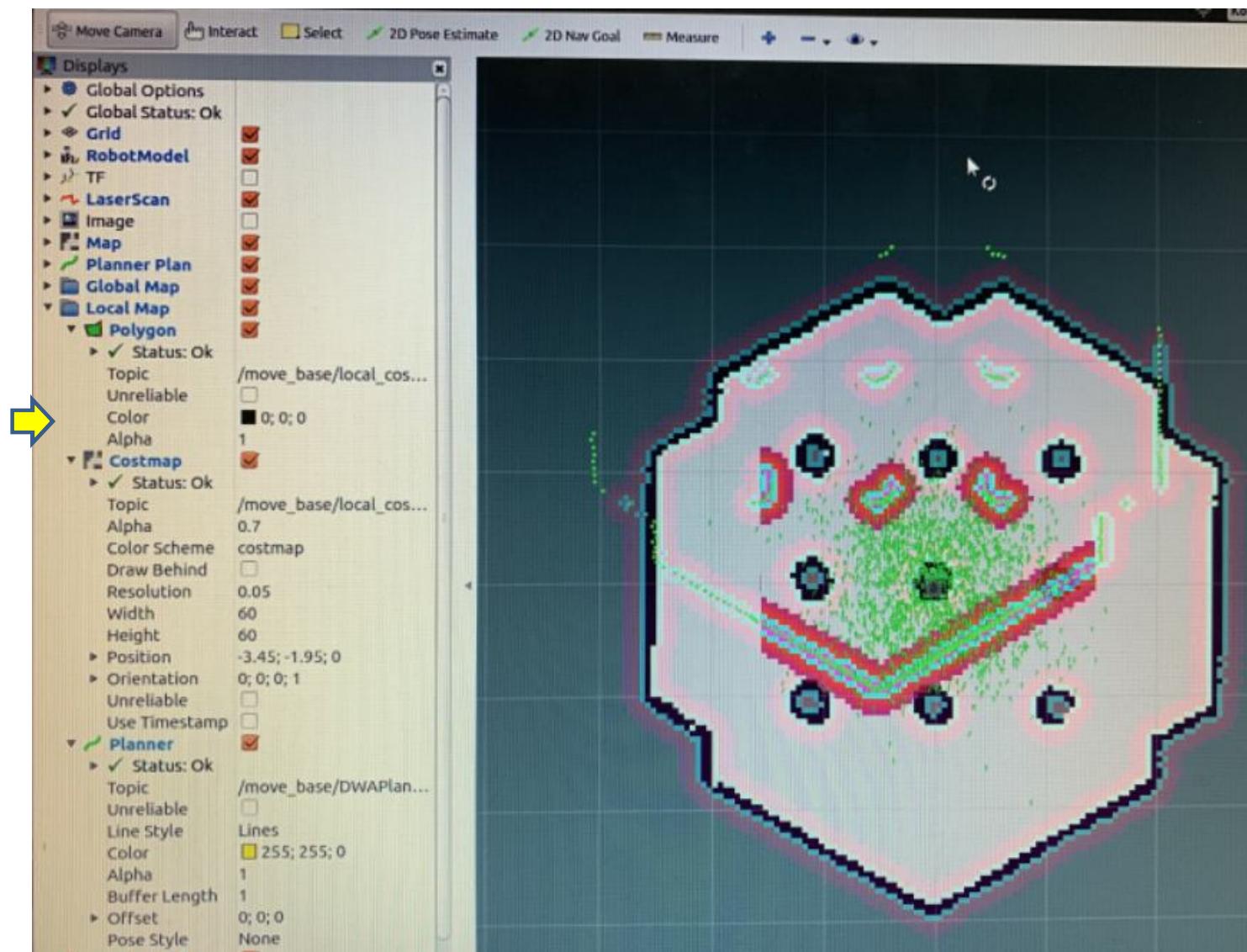
turtlebot을 제자리 회전하면

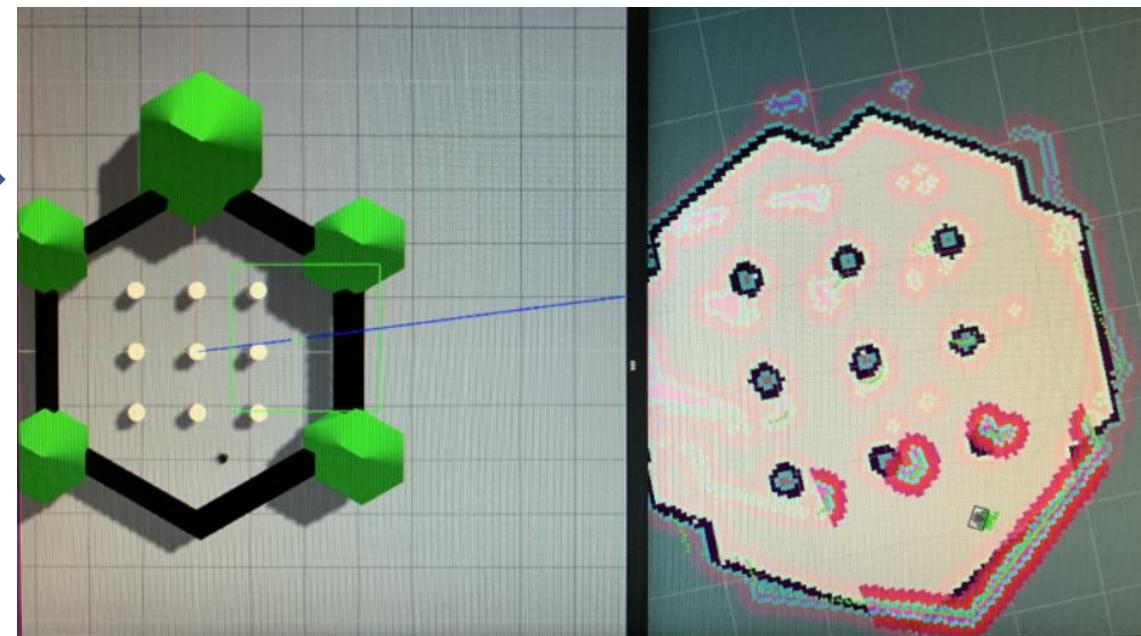
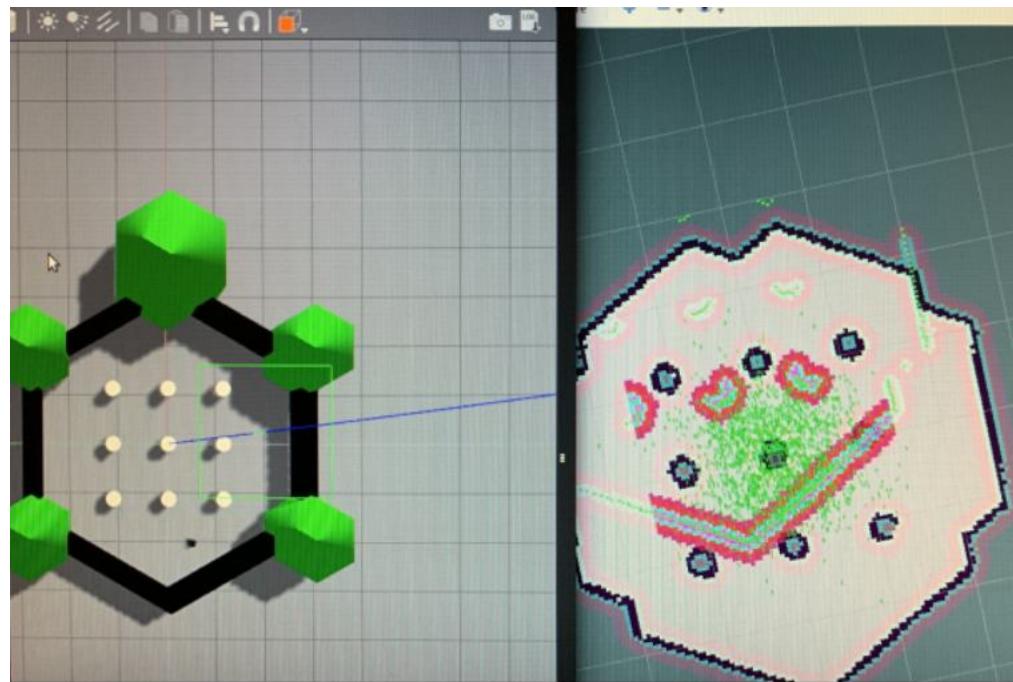
map과 장애물이 일치됨 (효과적)

→ 일치시킨 후 반드시

turtlebot3_teleop_key 를

CTL + C 로 중단시키기





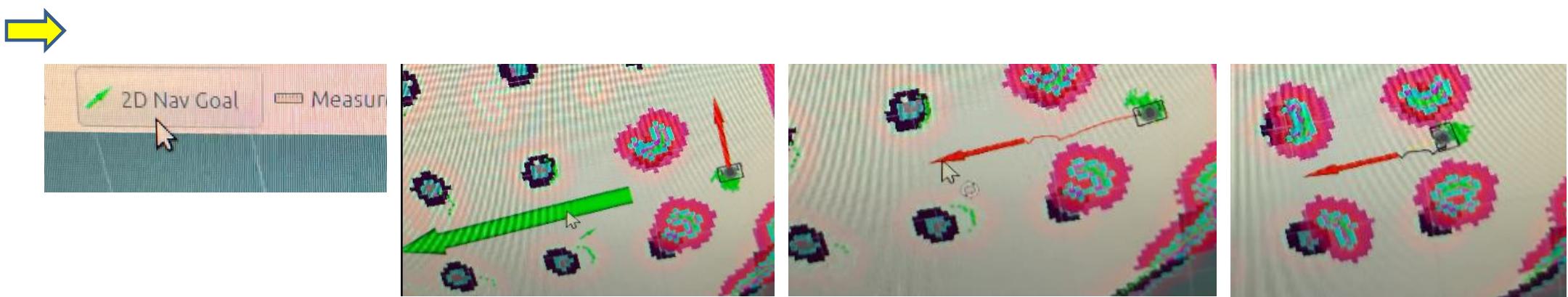
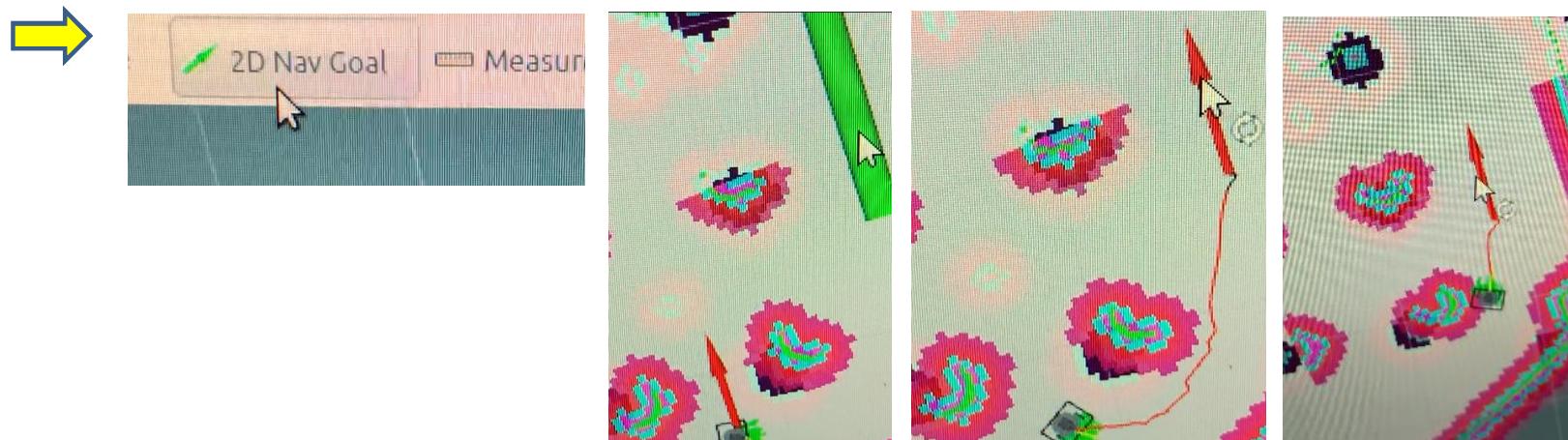
→ 일치시킨 후 반드시
turtlebot3_teleop_key 를
CTL + C 로 중단시키기

→ Navigation

: 2D Nav Goal를 누르고 목표를 향해

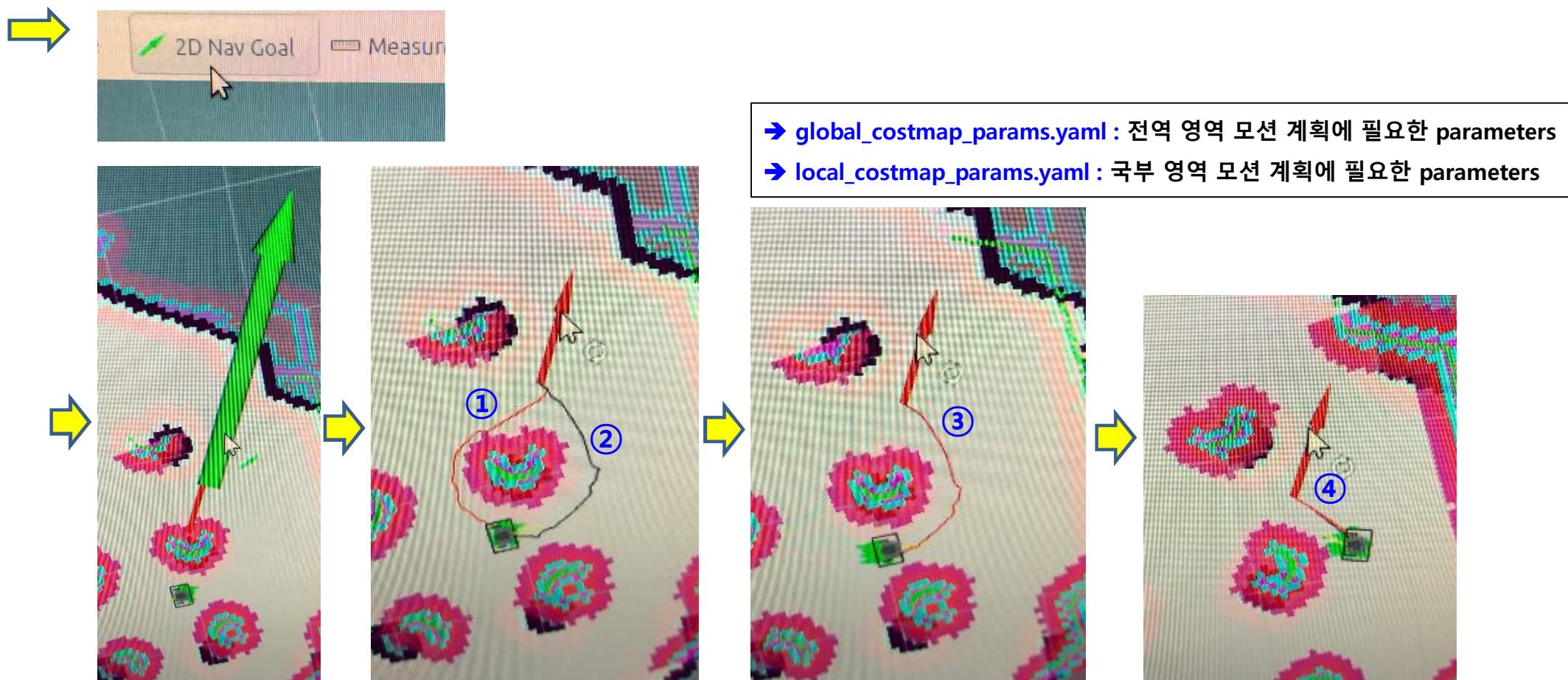
마우스를 통해 시작점과 끝점을 drag 하여 초록색 큰 화살표를 그리기

: 그후 자동으로 빨간색 화살표와 연결선이 생성되고 turtlebot 이동함



→ Navigation

- : 2D Nav Goal를 누르고 목표를 향해 마우스를 통해 시작점과 끝점을 drag 하여 초록색 큰 화살표를 그리기
- : 그후 자동으로 빨간색 화살표와 연결선이 생성되고 turtlebot 이동함
- ➔ 이동시 경로를 수시로 최적화하여 변경하고, 변경된 경로에 따라 이동함



감사합니다