Spring, 2021

지능로봇

SLAM AND NAVIGATION

메카트로닉스공학과

한국산업기술대학교

PRACTICE

- SYSTEM INTERFACE
- Navigation

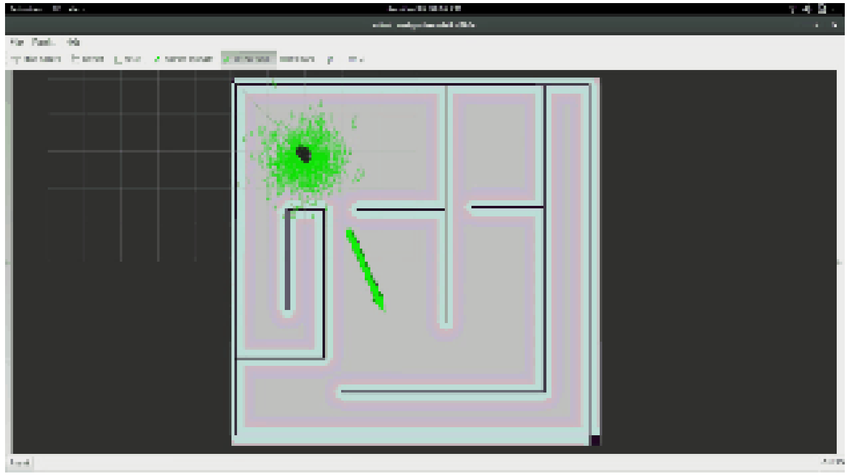
NAVIGATION

- WHAT IS NAVIGATION
- WHY NAVIGATION
- PREPARE FOR NAVIGATION
- GRAPH OF NAVIGATION

01. NAVIGATION

- WHAT IS NAVIGATION

Navigation은 모바일 로봇이 어느 지점에서 목표 지점으로 이동할 때, **로봇의 운동을 제어**하고 **운동 과정을 관측**하는 데에 중점을 둔 연구 분야이다.



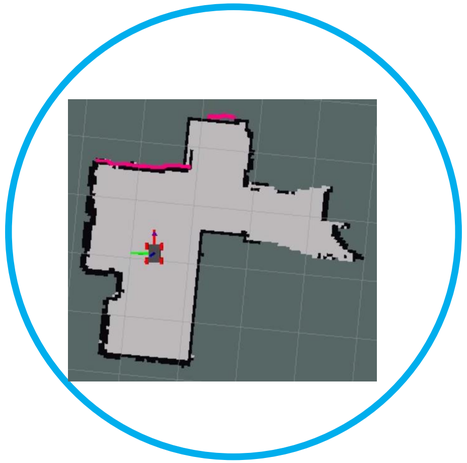
- WHY NAVIGATION?

어느 지점에서 목표 지점으로 이동할 때, 로봇이 선택할 수 있는 **루트는 무한**하다. 또한 이동간 **장애물이 존재**할 수도 있다.
Navigation을 이용하면 **안전하고 빠른 루트**로 **장애물을 회피**하여 갈 수 있다.

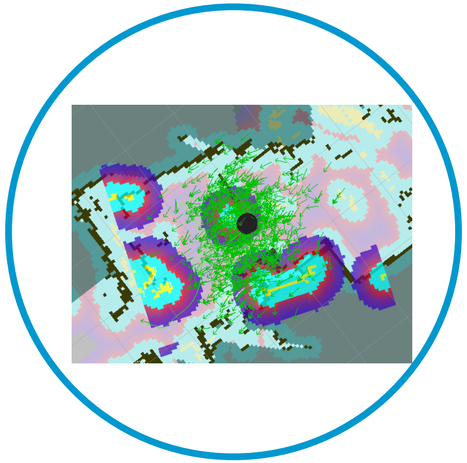
01. NAVIGATION

- PREPARE FOR NAVIGATION

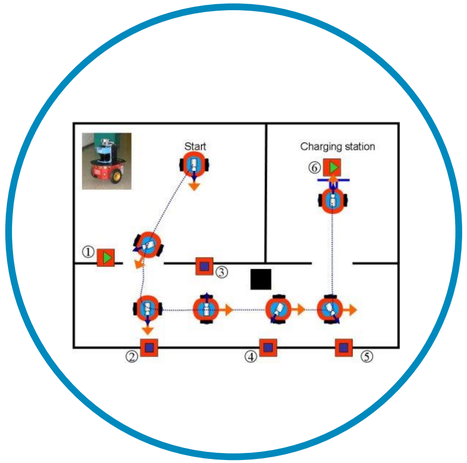
Navigation은 지도와 로봇의 센서 등을 이용하여 현재 위치로부터 지도상에 지정된 목적지까지 이동하게 된다.
그 순서로는 아래와 같다.



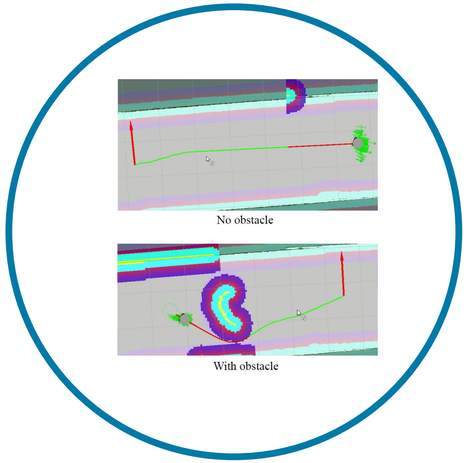
1) 센싱
지도상에서 로봇은 센서를 이용하여 자신의 **오도메트리 정보를 갱신**하면서 거리 센서가 장착된 위치로부터 **장애물과의 거리를 계측**한다.



2) 위치추정
기존에 작성해둔 지도상에 현재 로봇이 어디에 있는지 **위치 추정**을 하게 된다.



3) 모션계획
현재 위치로부터 지도상에 지정받은 목표 지점까지 **이동궤적을 생성**한다.



4) 이동 / 장애물 회피
모션계획에서 작성된 이동 궤적을 따라서 **목적지까지 이동**한다. 갑자기 나타난 **장애물을 회피**하며 이동한다.

01. NAVIGATION

- COSTMAP

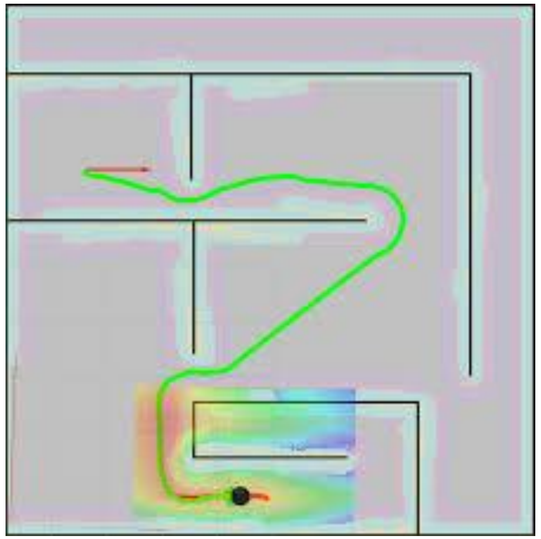
Navigation에서는 **1)로봇 위치**, **2)센서 위치**, **3)장애물 위치 정보**, 그리고 SLAM의 결과물로 얻은 **4)고정 지도(static map)** 4가지 요소를 기반으로 **장애물 영역**, **장애물과 충돌이 예상되는 영역**, **로봇이 이동 가능한 영역**을 계산하게 되는데 이것을 **costmap** 이라고 함.

- GLOBAL COSTMAP

global_costmap은 **전역 이동 경로 계획**에서 **고정 지도의 전체 영역을 대상으로 이동 계획**을 수립하는데 이 때의 출력물을 말함.

- LOCAL COSTMAP

local_costmap은 **국부 이동 경로 계획**에서 로봇을 중심으로 **일부 한정된 영역에 대해 이동 계획**할 때나 **장애물 회피**에 사용되는 지도를 말함.



000~000	로봇이 자유롭게 이동 가능한 자유 영역 (free area)
001~127	충돌하지 않는 영역
128~252	충돌 가능성이 있는 영역
253~254	충돌 영역
255~255	로봇이 이동 불가능한 점유 영역 (occupied area)

01. NAVIGATION

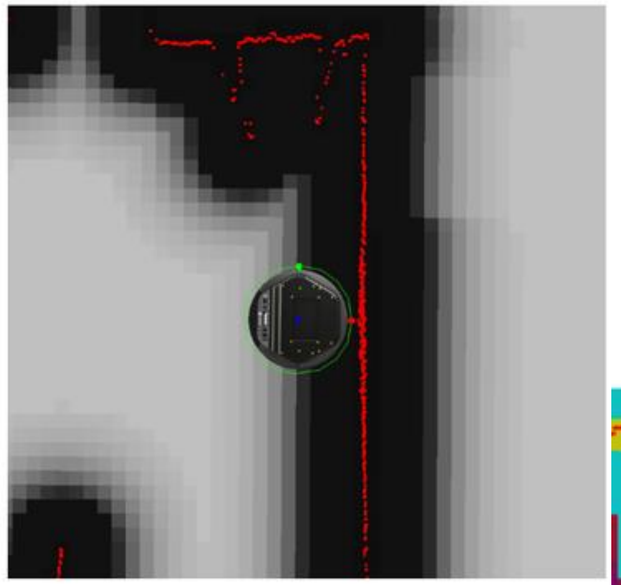
- LOCAL COSTMAP DISPLAY

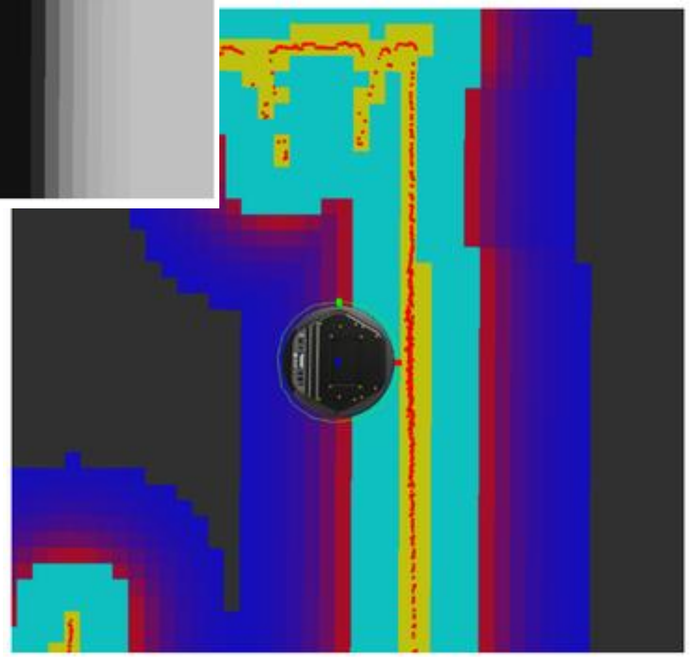
GREY SCALE

- 가운데에 터틀봇이 있고 그 주위로 **녹색 테두리**가 존재.
  이 녹색 라인이 실제 벽에 부딛치면 **로봇은 충돌**.
- **빨간색**은 레이저 센서에서 얻은 거리 값으로 **장애물을 표시**.
- **검은색**으로 갈수록 **충돌 가능성이 높은 위치**.

COLOR

- **노란색 셀**이 **실제 장애물**.
- **하늘색 영역**은 로봇 중심 위치가 이 영역으로 오게 되면 **충돌하게 되는 지점**. **굵은 빨간색 픽셀**로 **경계선 표시**.
- **파랑색**은 **안전한 지역** 의미.





01. NAVIGATION

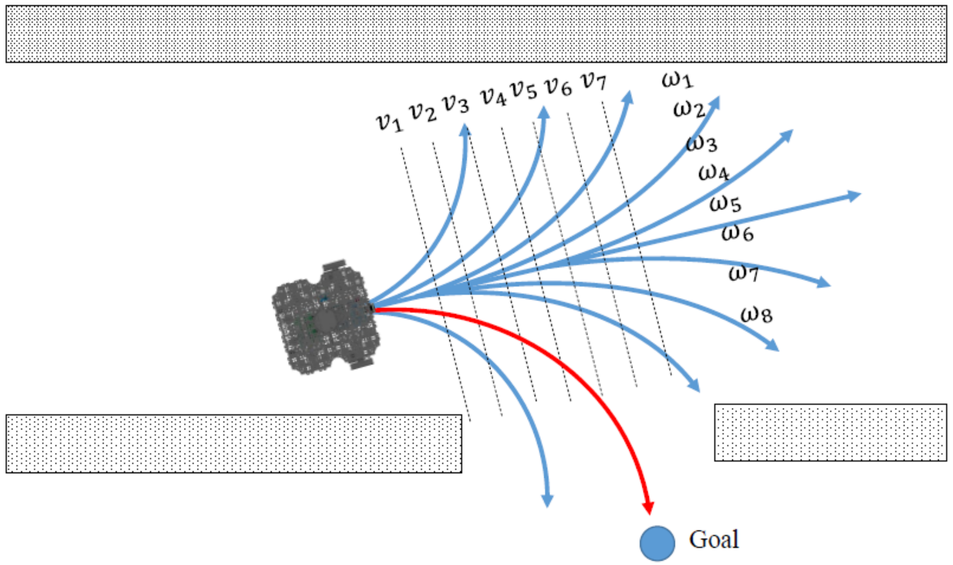
- DYNAMIC WINDOW APPROACH

- 로봇의 속도 탐색 영역(velocity search space)에서 로봇과 **충돌 가능한 장애물을 회피하면서 목표점까지 빠르게 다다를 수 있는 속도를 선택하는 방법.**

- $G(v,\omega)=\sigma(\alpha\cdot heading(v,\omega)+\beta\cdot dist(v,\omega)+\gamma\cdot velocity(v,\omega))$

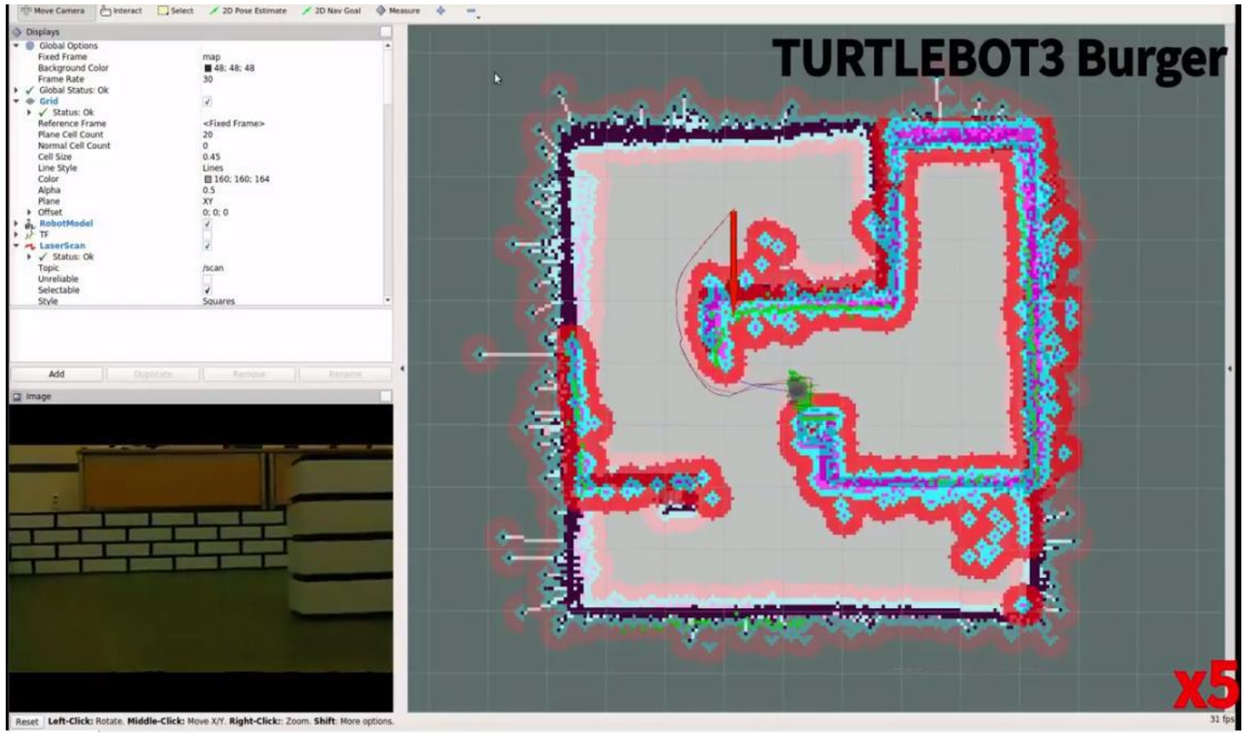
- 목적함수 G는 로봇의 **방향, 속도, 충돌을 고려**하여, 목적함수가 최대가 되는 속도 v,ω를 구하게 됨.

- α, β, γ 인수 값에 따라 방향에 치중할지 거리에 치중할지 속도에 치중할지 결정.



01. NAVIGATION

- NAVIGATION VIDEO



https://youtu.be/VYlMywwYALU

01. NAVIGATION

- GRAPH OF NAVIGATION (TURTLEBOT)

01
- **오도메트리 (“odom”, nav_msgs/Odometry)**
  - **로봇의 현재 위치 및 속도** 등의 정보를 받아
  국부 이동 경로를 생성하거나 장애물 회피 등에 사용된다.

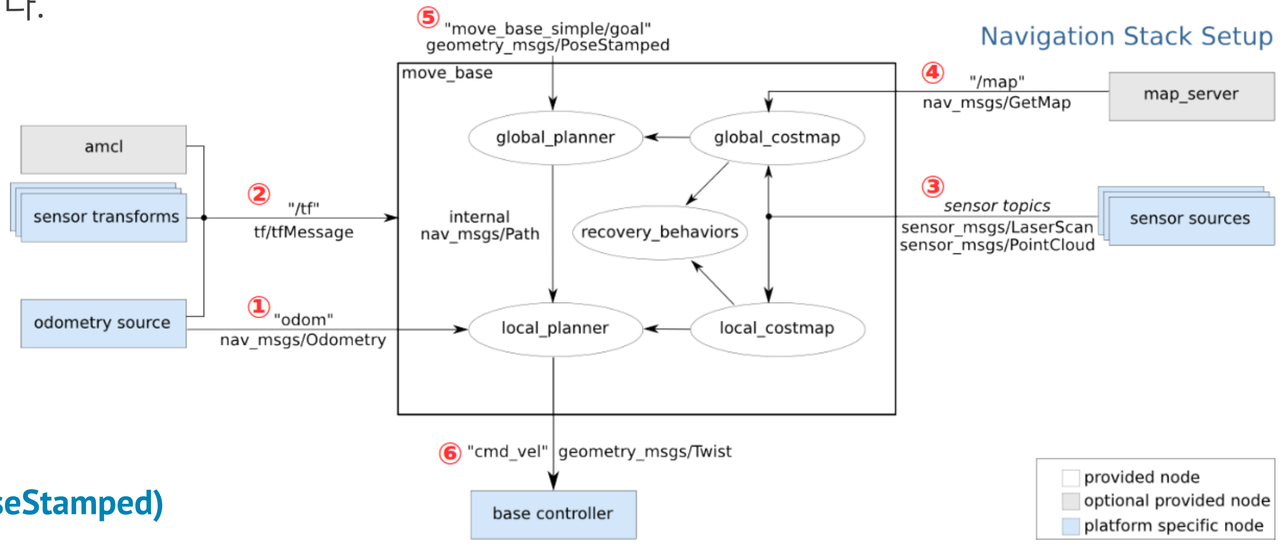
02
- **상대 위치 변환 (“/tf”, tf/tfMessage)**
  - 로봇의 위치로 부터 **변환된 센서의 위치**를 획득하여
  이동 경로 계획을 수행하게 된다.

03
- **거리 센서 (sensor topics, sensor_msg/LaserScan)**
  - LRF 센서로부터 측정된 **거리 값**을 의미한다.
  로봇의 현재 위치 추정, 로봇의 모션 계획에 사용된다.

04
- **지도 (“/map”, nav_msgs/GetMap)**
  - SLAM에서 작성된 “map.pgm”과 “map.yaml”을
  map_server 패키지를 이용해 **map을 발행**한다.

05
- **목표 좌표 (“move_base_simple/goal”, geometry_msgs/PoseStamped)**
  - 유저가 직접 지정하는 **로봇의 목표 좌표**이다.
  목표 좌표는 2차원 상에 x, y, θ 로 구성되어 있다.

06
- **속도 명령 (“cmd_vel”, geometry_msgs/Twist)**
  - 최종적으로 계획된 이동 궤적에 따라 **로봇을 움직이는 속도 명령을 발행**하는 것으로
  로봇은 목적지까지 이동하게 된다.



02. PRACTICE

- SYSTEM INTERFACE (TURTLEBOT)

- Remote PC와 Robot PC가 같은 WIFI 환경을 갖도록 하여 **서로 통신이 가능하도록** 함

WIFI 공유기

LRF sensor

- Laser Range Finder
- **거리센서 역할**

SERIAL

Robot PC (raspbetty pi)

- LRF, Encoder, 3축 자이로 등의 **센서 값을 받거나 모터를 동작** 시킴으로써 **SLAM 또는 Navigation을 진행**

WIFI

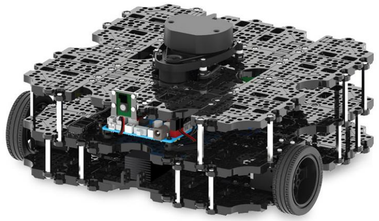
WIFI

Remote PC

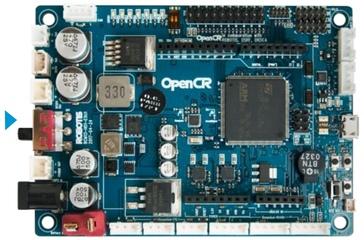
- **ROS MASTER** 역할
- keyop 노드를 이용해 로봇을 **원격조종**
- **Rviz**를 이용해 Robot PC에서 받은 정보들을 **Display**

SERIAL

Turtlebot3



Embedded Board



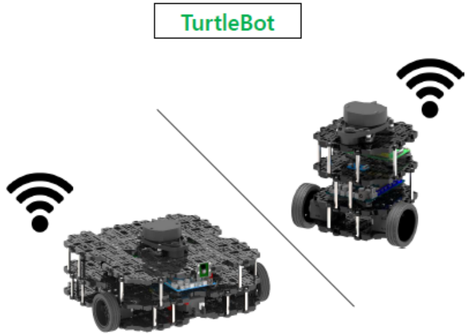
- Robot PC에 **센서 값들을 전달** (3축 자이로, 바퀴 엔코더 등)
- Robot PC로 부터 명령을 받아 **모터를 구동**

02. PRACTICE

- Turtulebot3 개발환경 (네트워크)

- **Laptop 네트워크 설정**

1. Turtlebot3와 같은 wifi로 설정
2. Terminal에 ‘ifconfig’를 입력하여 ip address를 획득
3. Terminal에 ‘cd ~’를 입력하여 home으로 이동
4. Terminal에 ‘sudo nano .bashrc’를 입력
   (.bashrc : Terminal이 켜질 때 작동될 내용들)
5. 맨 밑으로 이동하여 다음과 같이 입력 후 저장 (‘ctrl’+’x’)
   ROS_MASTER_URI=http://IP_OF_REMOTE_PC:11311
   ROS_HOSTNAME = IP_OF_REMOTE_PC
6. source ~/.bashrc (.bashrc에 갱신된 내용을 실행시킴)



ROS_MASTER_URI = http://IP_OF_REMOTE_PC:11311
ROS_HOSTNAME = IP_OF_TURTLEBOT

ROS_MASTER_URI = http://IP_OF_REMOTE_PC:11311
ROS_HOSTNAME = IP_OF_REMOTE_PC

```
# labtop setting
export ROS_MASTER_URI=http://192.168.0.101:11311
export ROS_HOSTNAME=192.168.0.101
```

02. PRACTICE

- Turtulebot3 개발환경 (네트워크)

- **Turtlebot3 네트워크 설정**

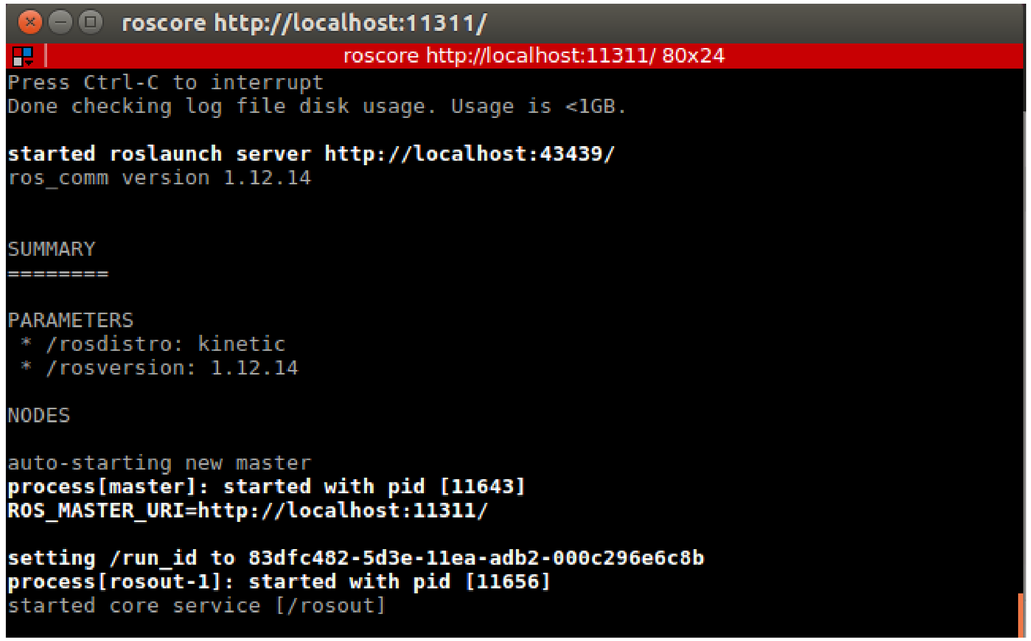
1. Laptop에서 ssh pi@IP_OF_TURTLEBOT 입력
   (Turtlebot3의 라즈베리파이에 원격접속)
2. Password 입력 : turtlebot
3. Terminal에 ‘ifconfig’를 입력하여 ip address를 획득
4. Terminal에 ‘cd ~’를 입력하여 home으로 이동
5. Terminal에 ‘sudo nano .bashrc’를 입력
   (.bashrc : Terminal이 켜질 때 작동될 내용들)
6. 맨 밑으로 이동하여 다음과 같이 입력 후 저장
   ROS_MASTER_URI=http://IP_OF_REMOTE_PC:11311
   ROS_HOSTNAME = IP_OF_TURTLEBOT
7. source ~/.bashrc (.bashrc에 갱신된 내용을 실행시킴)

```
#turtlebot3 setting
export ROS_MASTER_URI=http://192.168.0.101:11311
export ROS_HOSTNAME=192.168.0.115
```

02. PRACTICE

- Navigation [학생]

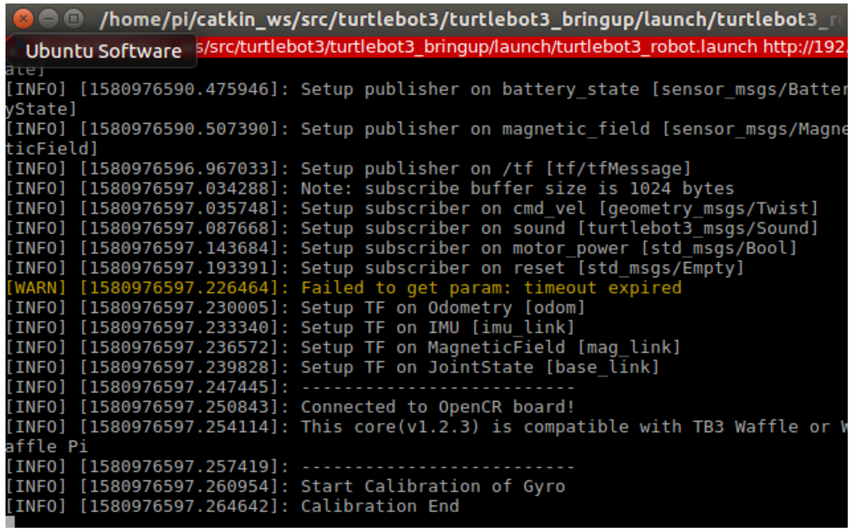
- [Remote PC] ROS Master를 실행
  $ roscore



02. PRACTICE

- Navigation [학생]

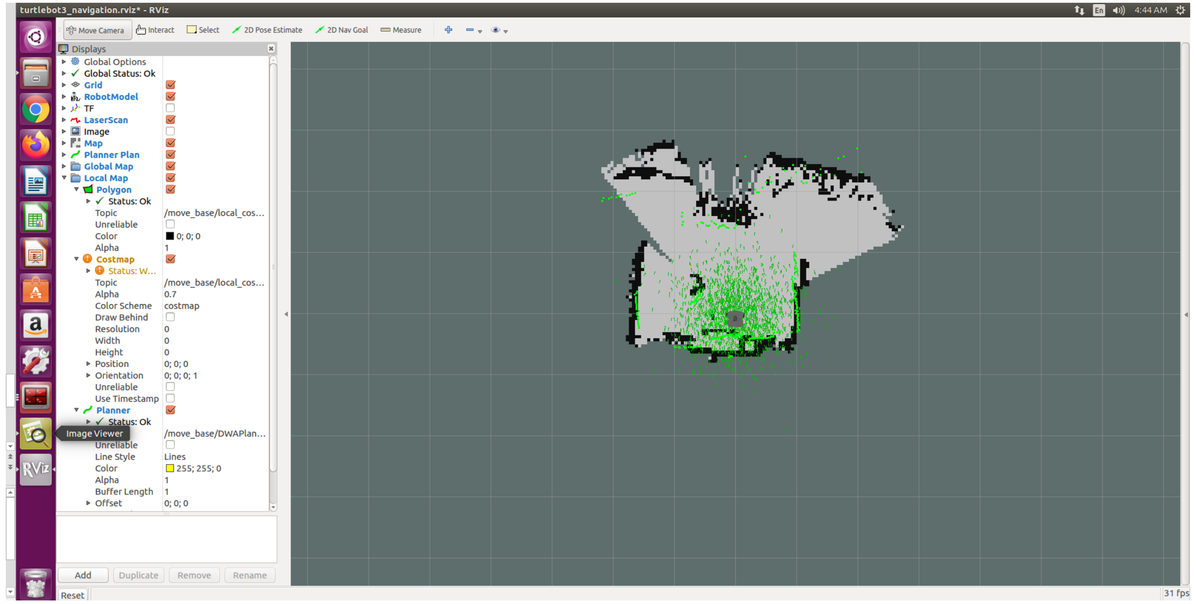
- [TurtleBot] 터틀봇에 원격으로 접속하여 운동을 위한 노드를 실행시킴
  $ ssh pi@IP_OF_TURTLEBOT
  (password : turtlebot)
  $ roslaunch turtlebot3_bringup turtlebot3_robot.launch --screen



02. PRACTICE

- Navigation [학생]

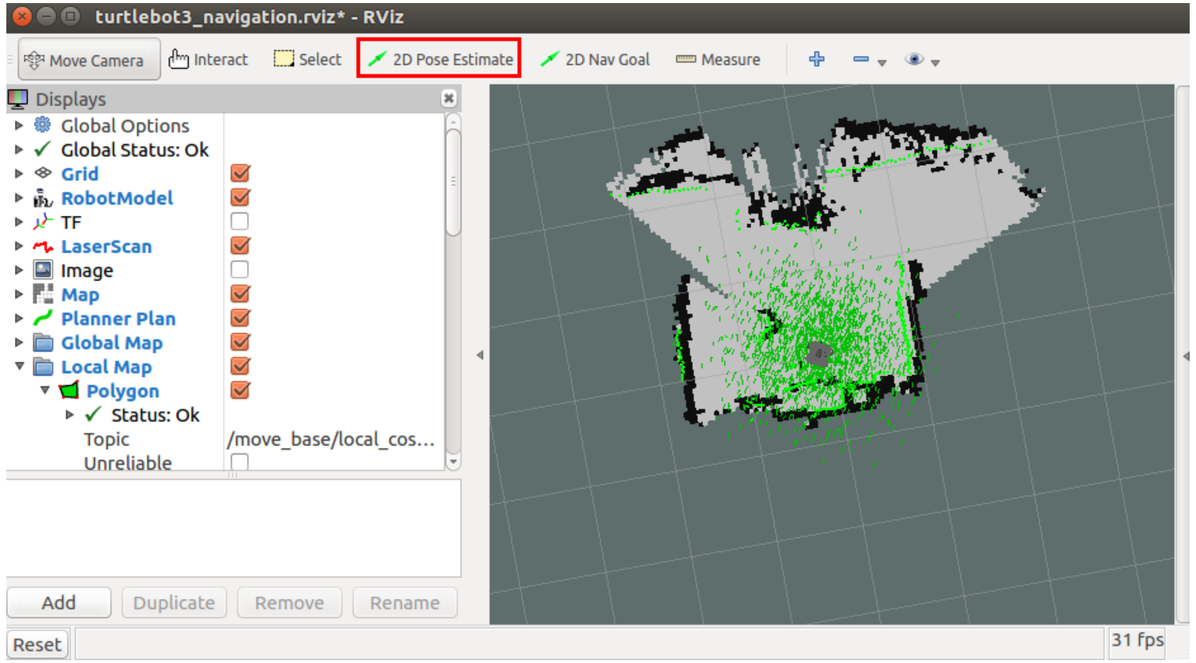
- [Remote PC] 로봇모델을 설정하고 navigation을 실행
  $ export TURTLEBOT3_MODEL=waffle_pi
  $ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml



02. PRACTICE

- Navigation [학생]

- [Remote PC] 2D Pose Estimate 버튼을 이용하여 로봇의 초기 위치를 대략적으로 지정해줌



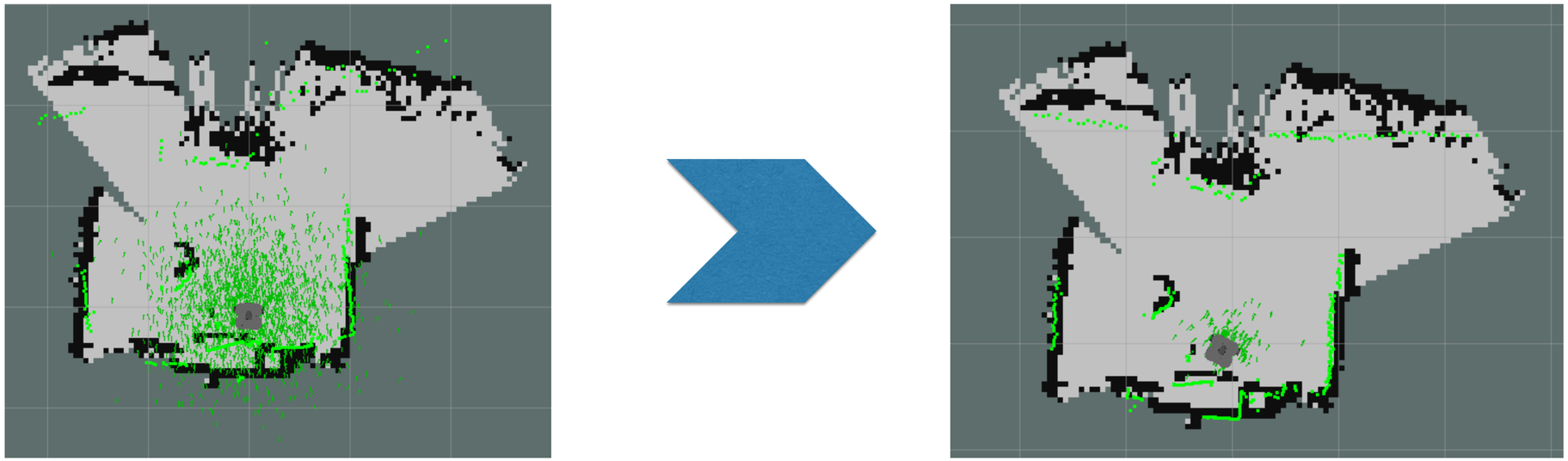
02. PRACTICE

- Navigation [학생]

- [Remote PC] teleop_key 노드를 이용해 키보드로 터틀봇을 조작하며 파티클들을 수렴시킴

  $ export TURTLEBOT3_MODEL=waffle_pi

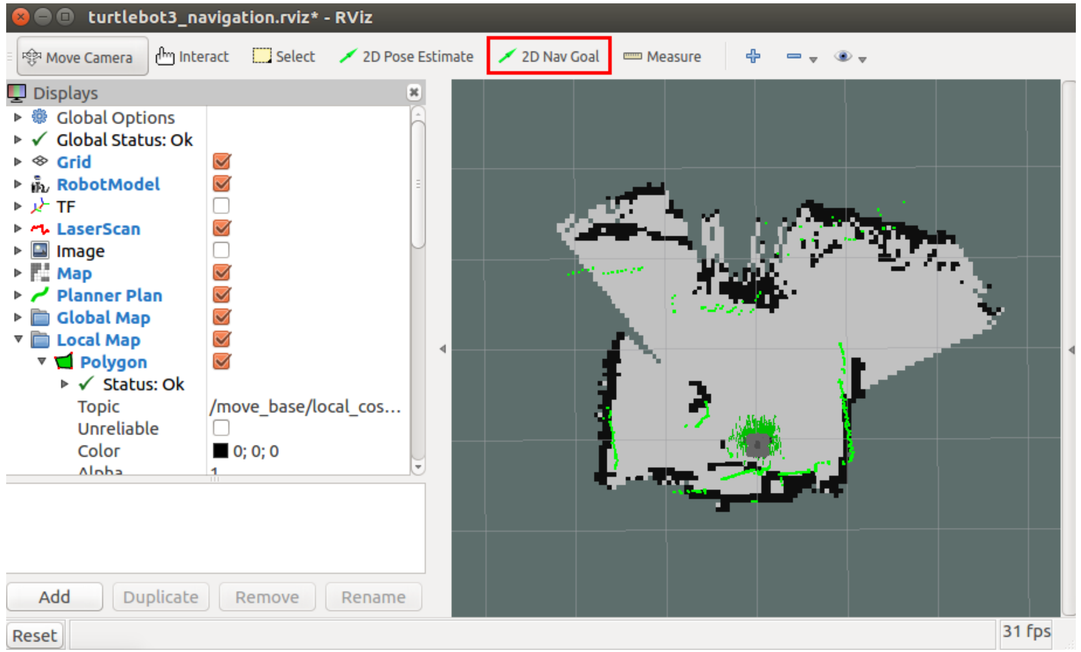
  $ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch --screen



02. PRACTICE

- Navigation [학생]

- [Remote PC] particle filter가 정밀해지면 ‘ctrl’ + ‘c’를 눌러 teleop_key 노드를 종료시키고 2D Nav Goal 버튼을 클릭하여 특정 포지션으로 이동시킴



02. PRACTICE

- Navigation [학생]

- 모든 작업이 끝나면 ‘ctrl’ + ‘c’ 를 눌러 모든 노드를 종료시킨다.

THANK YOU

SOURCES

[1] 로봇 운영체제 ROS 강좌, Pyo yunseok , 2015.3.2,
https://cafe.naver.com/openrt/2360

[2] SLAM, jacobyu, 2018, https://steemit.com/kr-dev/@jacobyu/54qama