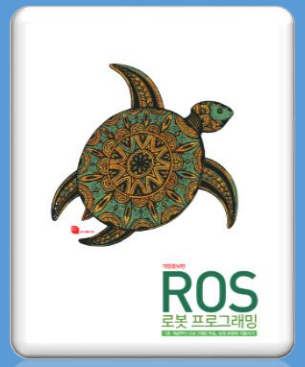
SLAM과 내비게이션

ROBOTIS

Open Source Team

Yoonseok Pyo

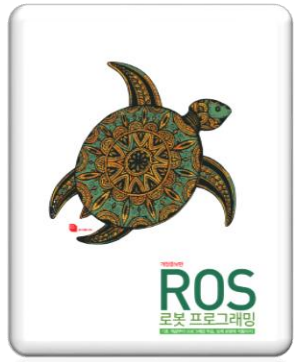


온라인강좌

YouTube Subscribe

교재
P. 327~377

Contents



온라인강좌

YouTube Subscribe

교재
P. 327~377

자~ 이번 주제는

SLAM, Navigation

입니다!

Simultaneous Localization And Mapping
&
Navigation

뭔 말이야?

—— ;;

동시적 위치 추정 및 지도 작성
&
차량 자동 항법 장치

뭐야? OTL...
더 어려워 보이잖아!
——;;

좀~ 쉽게 갑시다!

길 찾기

어때요?

길…

길「명사」

1. 사람이나 동물 또는 자동차 따위가 지나갈 수 있게 땅 위에 낸 일정한 너비의 공간.
2. 물 위나 공중에서 일정하게 다니는 곳.
3. 걷거나 탈것을 타고 어느 곳으로 가는 노정(路程).

-국립국어원 표준국어대사전-

길 찾기...

길「명사」

1. 사람이나 동물 또는 자동차 따위가 지나갈 수 있게 땅 위에 낸 일정한 너비의 공간.
2. 물 위나 공중에서 일정하게 다니는 곳.
3. 걷거나 탈것을 타고 어느 곳으로 가는 노정(路程).

-국립국어원 표준국어대사전-

여행의 오랜 동반자

나침반과 지도

Ciaran Griffin, http://www.gettyimages.com

나침반도 없고
지도도 없다면?

http://www.pinterest.com/jerodor/cagaduisimo/

나는 여기에 있다.

- 해, 달, 별의 위치만으로 떠나는 여행자
- 중국 4대 발명 나침반
- 나침반 진화
  - 자기 나침반
  - 전륜 나침반
  - GPS
- 지도

Big Dipper, by Magnus Manske, Public Domain

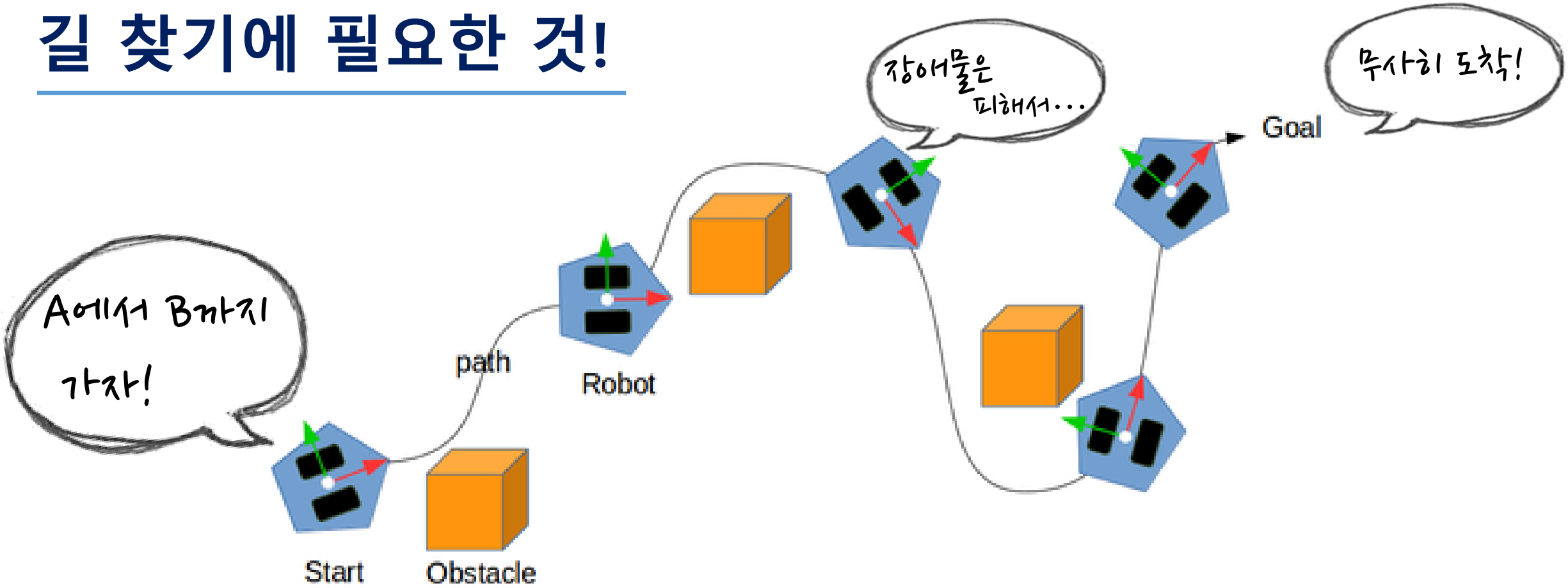
pixabay.com, CC0

상상해 보세요!
어둠 속 길 찾기

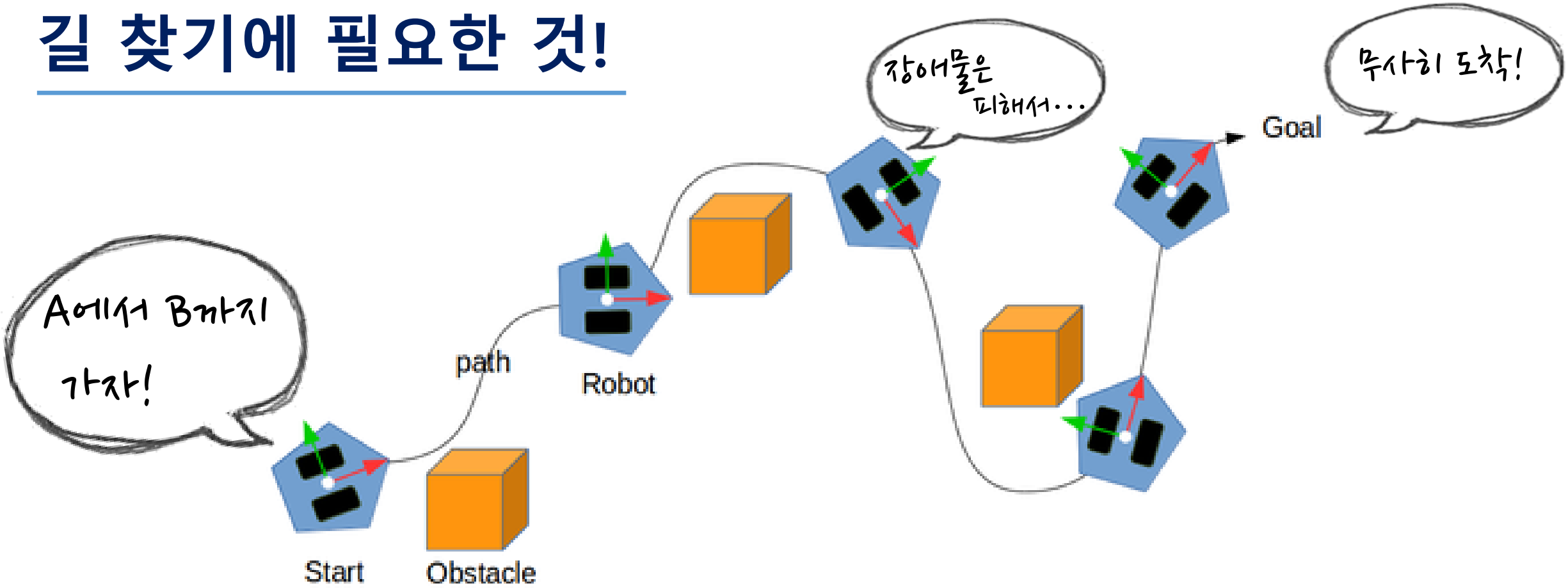
로봇의 길 찾기

(이제부터는 성실히 풀어가 볼게요.)

길 찾기에 필요한 것!



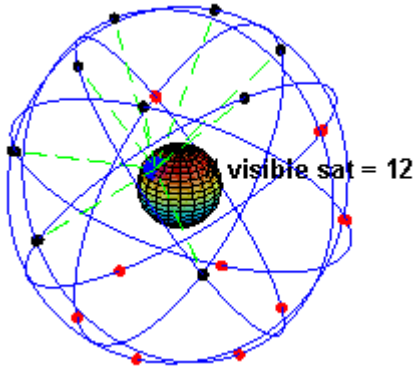
길 찾기에 필요한 것!



① **위치**: 로봇의 위치 계측/추정하는 기능
② **센싱**: 벽, 물체 등의 장애물의 계측하는 기능
③ **지도**: 길과 장애물 정보가 담긴 지도
④ **경로**: 목적지까지 최적 경로를 계산하고 주행하는 기능

① 위치: 로봇의 위치 계측/추정하는 기능

- GPS (Global Positioning System)



  - 오차
  - 날씨
  - 실외

- Indoor Positioning Sensor
  - Landmark (Color, IR Camera)
  - Indoor GPS
  - WiFi SLAM
  - Beacon

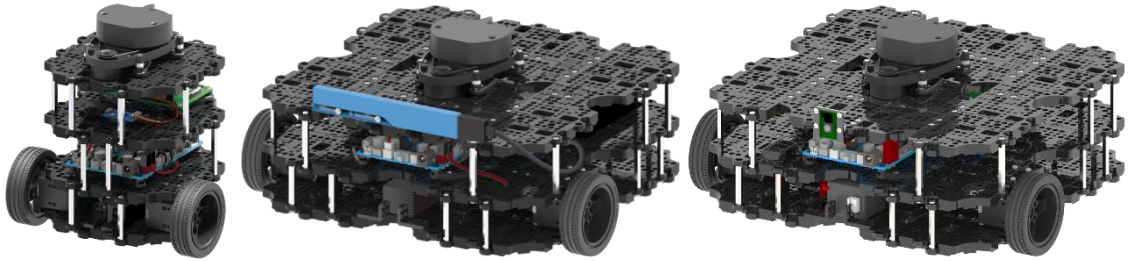
Estimote (Beacon)

StarGazer

Vicon MX

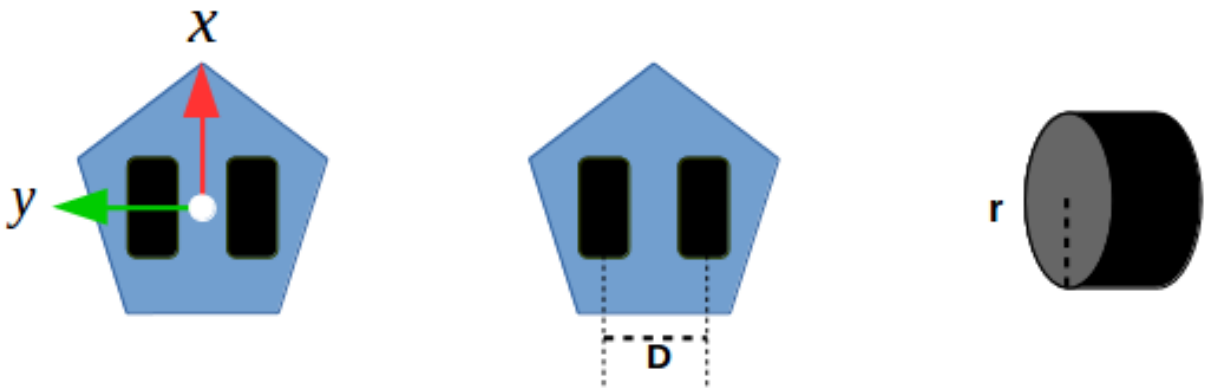
① 위치: 로봇의 위치 계측/추정하는 기능

- 추측 항법(dead reckoning, 데드레커닝)
  - 양 바퀴 축의 회전 값을 이용
  - 이동 거리와 회전 값을 계산, 위치 측정
  - 바닥 슬립, 기계적, 누적 오차 발생
  - IMU 등의 관성 센서, 필터로 위치 보상
  - 칼만필터 시리즈…



TurtleBot 3

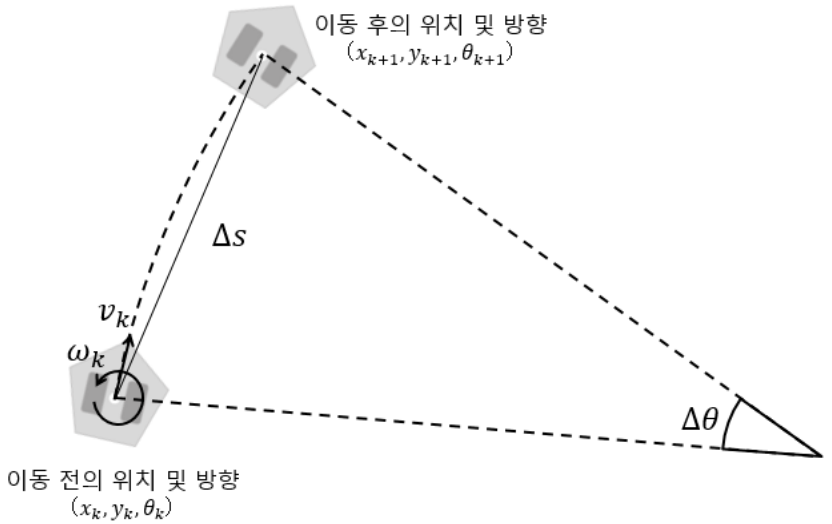
- 필요한 정보
  - 양 바퀴 축의 엔코더 값 E
    (모터 축인 경우 기어비로 재계산)
  - 바퀴 간 거리 D
  - 바퀴 반지름 r

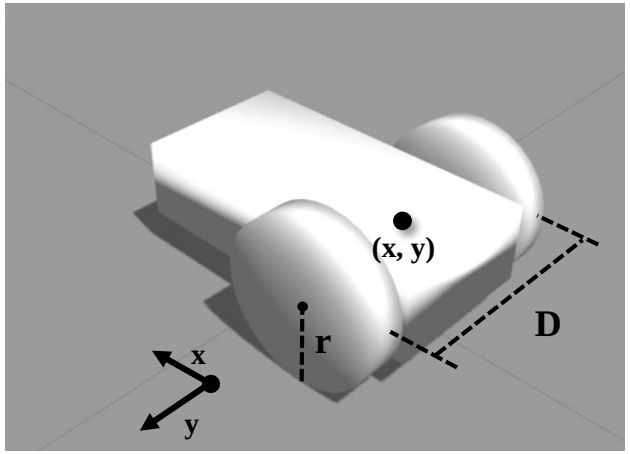


① 위치: 로봇의 위치 계측/추정하는 기능

- 데드레커닝 계산
  - 선속도(linear velocity: v)
  - 각속도(angular velocity: w)

- Runge-Kutta 공식 이용
  - 이동한 위치의 근사 값 x, y
  - 회전 각도 θ





$$v_l = \frac{(E_l c - E_l p)}{T_e} \cdot \frac{\pi}{180} \text{ (radian/sec)}$$

$$v_r = \frac{(E_r c - E_r p)}{T_e} \cdot \frac{\pi}{180} \text{ (radian/sec)}$$

$$V_l = v_l \cdot r \text{ (meter/sec)}$$

$$V_r = v_r \cdot r \text{ (meter/sec)}$$

$$v_k = \frac{(V_r + V_l)}{2} \text{ (meter/sec)}$$

$$\omega_k = \frac{(V_r - V_l)}{D} \text{ (radian/sec)}$$

$$\Delta s = v_k T_e \qquad \Delta\theta = \omega_k T_e$$

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

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

$$\theta_{(k+1)} = \theta_k + \Delta\theta$$

① 위치: 로봇의 위치 계측/추정하는 기능

```
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;
```

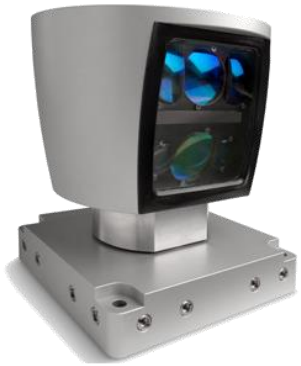
```
 delta_s     = WHEEL_RADIUS * (wheel_r + wheel_l) / 2.0;
  orientation = sensors.getOrientation();
  theta       = atan2f(orientation[1]*orientation[2] +
                       orientation[0]*orientation[3],
                       0.5f - orientation[2]*orientation[2] -
                       orientation[3]*orientation[3]);

  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;
}
```

turtlebot_core 예제 (calcOdometry 함수)

② 센싱: 벽, 물체 등의 장애물의 계측하는 기능

- 거리센서
  - LRF, 초음파센서, 적외선 거리센서(PSD)





- 비전센서
  - 스테레오 카메라, 모노 카메라, 전 방향 옴니 카메라

- Depth camera
  - SwissRanger, Kinect-2
  - RealSense, Kinect, Xtion, Carmine(PrimeSense), Astra









③ 지도: 길과 장애물 정보가 담긴 지도

- 로봇은 길을 찾아가기 위해 지도가 필요하다!

- 지도
  - 도로와 같은 기반 시설의 경우 디지털 지도 OK!
  - 병원, 카페, 회사, 가정집의 지도?
  - 탐사, 붕괴된 위험지역의 지도?

③ 지도: 길과 장애물 정보가 담긴 지도

- 로봇은 길을 찾아가기 위해 지도가 필요하다!
- 지도
  - 도로와 같은 기반 시설의 경우 디지털 지도 OK!
  - 병원, 카페, 회사, 가정집의 지도?
  - 탐사, 붕괴된 위험지역의 지도?

- **지도? 없으면 만들자!**

- **SLAM**

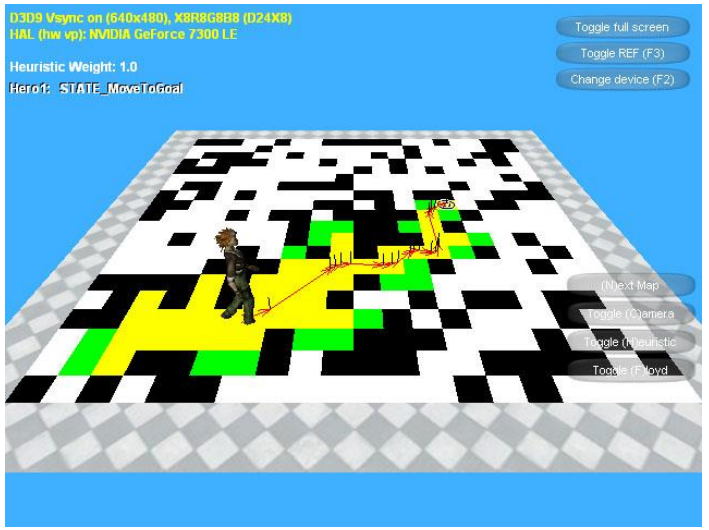
(Simultaneous Localization And Mapping)

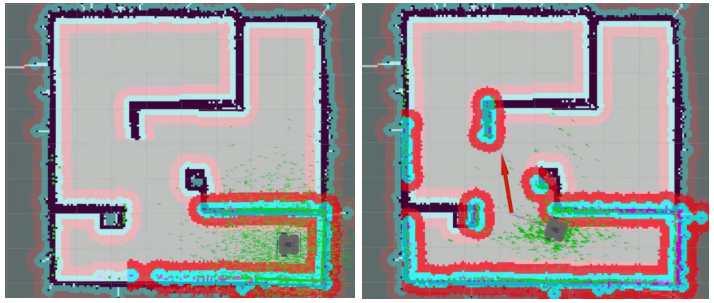
같이 여긴 어디? 지도 만들자

④ 경로: 목적지까지 최적 경로를 계산하고 주행하는 기능

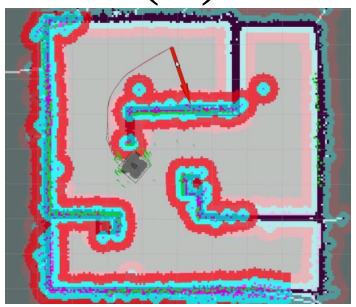
- 내비게이션(Navigation)
- 위치 추정 (Localization / Pose estimation)
- 경로 탐색/계획 (Path search and planning)

- Dynamic Window Approach (DWA)
- A* 알고리즘 (A Star)
- 포텐셜 장(Potential Field)
- 파티클 필터 (Particle Filter)
- 그래프 (Graph)

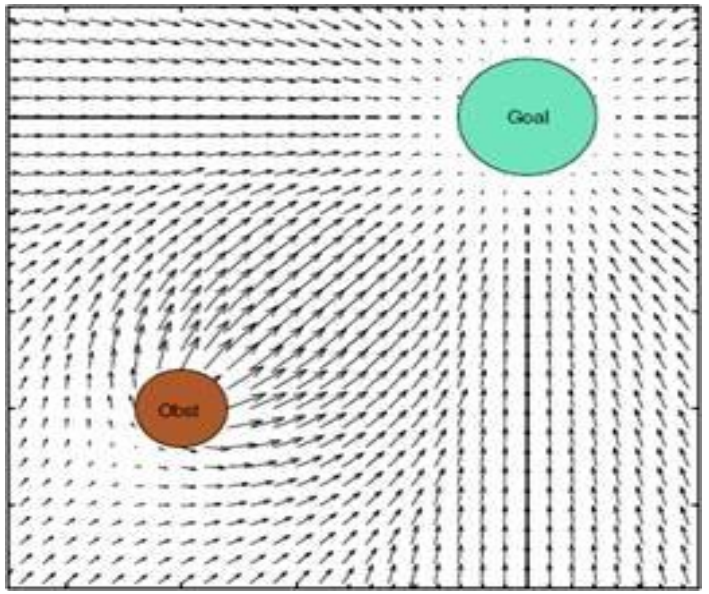



(t1)

(t2)


(t3)


(t4)



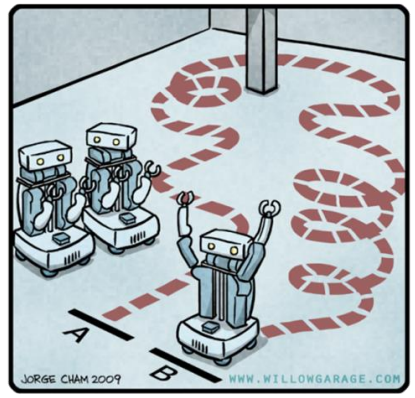
https://students.cs.byu.edu/~cs470ta, http://vimeo.com/3423169

① 위치

② 센싱

③ 지도

④ 경로

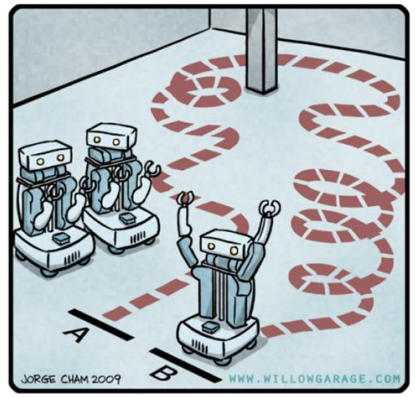


① 위치

② 센싱

③ 지도

④ 경로



위치+센싱 → **지도**

SLAM

위치+센싱+지도 → **경로**

Navigation

SLAM

Gmapping (*ROS Melodic 버전 부터는 gmapping 지원하지 않음, Google cartographer 를 추천함)

- OpenSLAM에 공개된 SLAM 의 한 종류, ROS에서 패키지로 제공
- 저자: G. Grisetti, C. Stachniss, W. Burgard
- 특징: Rao-Blackwellized 파티클 필터, 파티클 수 감소, 그리드 맵

- 하드웨어 제약 사항
  - **X, Y, Theta 속도 이동 명령**
    - **차동 구동형 모바일 로봇**(differential drive mobile robot)
    - **전 방향 이동 로봇** (omni-wheel robot)
  - **주행기록계 (Odometry)**
  - **계측 센서: 2차 평면 계측 가능 센서(LRF, LiDAR, Kinect, Xtion 등)**
  - 직사각형 및 원형의 로봇

https://www.openslam.org/gmapping.html

지도작성: Gmapping + TurtleBot3

- 소프트웨어 준비
  - http://emanual.robotis.com/docs/en/platform/turtlebot3/pc_setup/
  - http://emanual.robotis.com/docs/en/platform/turtlebot3/sbc_setup/
  - http://emanual.robotis.com/docs/en/platform/turtlebot3/opencr_setup/

- Turtlebot3 패키지
  - https://github.com/ROBOTIS-GIT/turtlebot3
  - https://github.com/ROBOTIS-GIT/turtlebot3_msgs
  - https://github.com/ROBOTIS-GIT/turtlebot3_simulations
  - https://github.com/ROBOTIS-GIT/turtlebot3_applications
  - https://github.com/ROBOTIS-GIT/turtlebot3_applications_msgs

지도작성: Gmapping + TurtleBot3

- http://emanual.robotis.com/docs/en/platform/turtlebot3/slam
- 마스터 실행 (Remote PC)

```
$ roscore
```

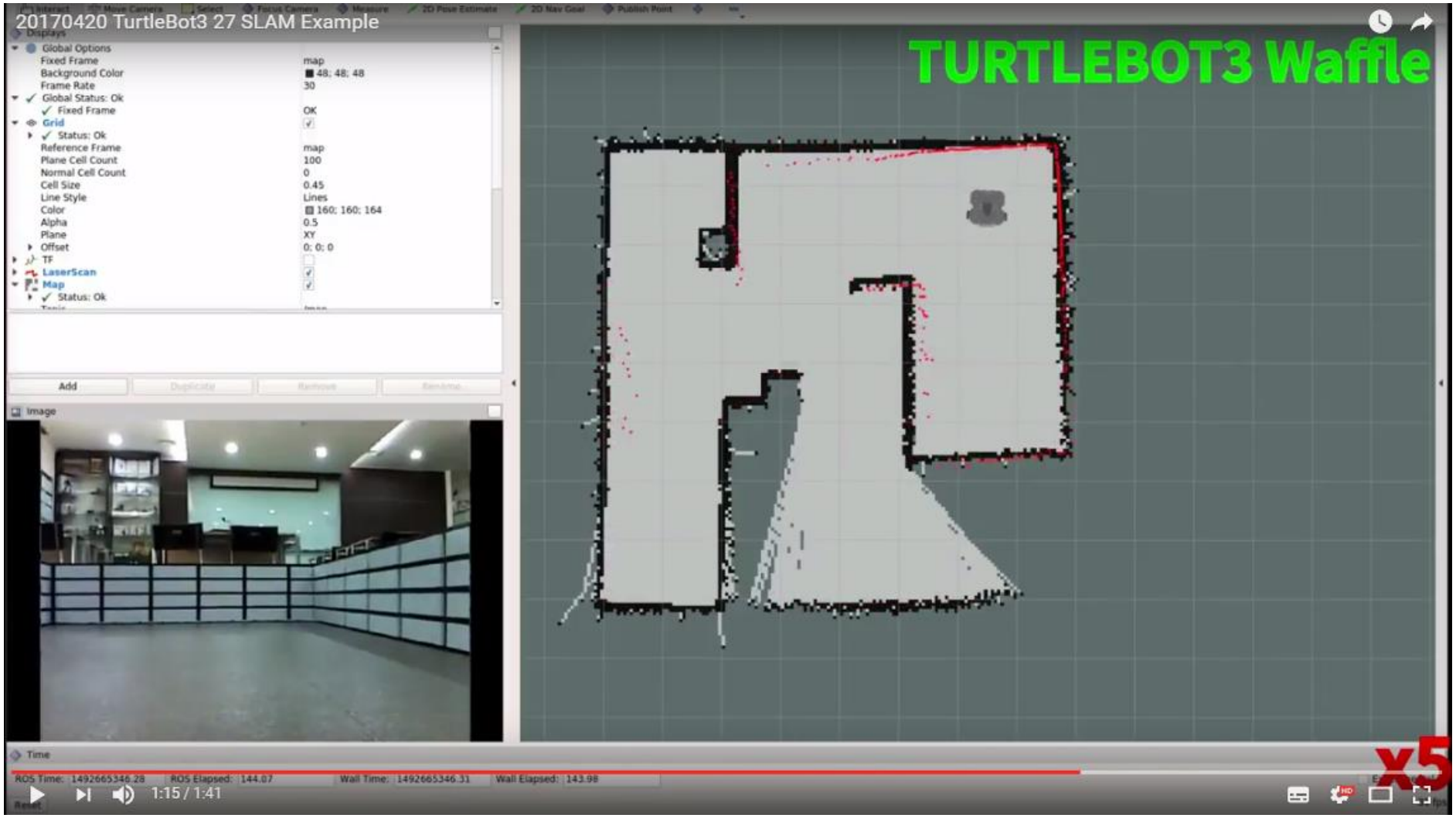
- 터틀봇 및 센서 구동 (SBC)

```
$ export TURTLEBOT3_MODEL=burger  (또는 waffle, waffle_pi)
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

- RViz, 터틀봇 원격 조종, 지도 작성 (Remote PC)

```
$ export TURTLEBOT3_MODEL=burger  (또는 waffle, waffle_pi)
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
$ rosrun map_server map_saver -f ~/map
```

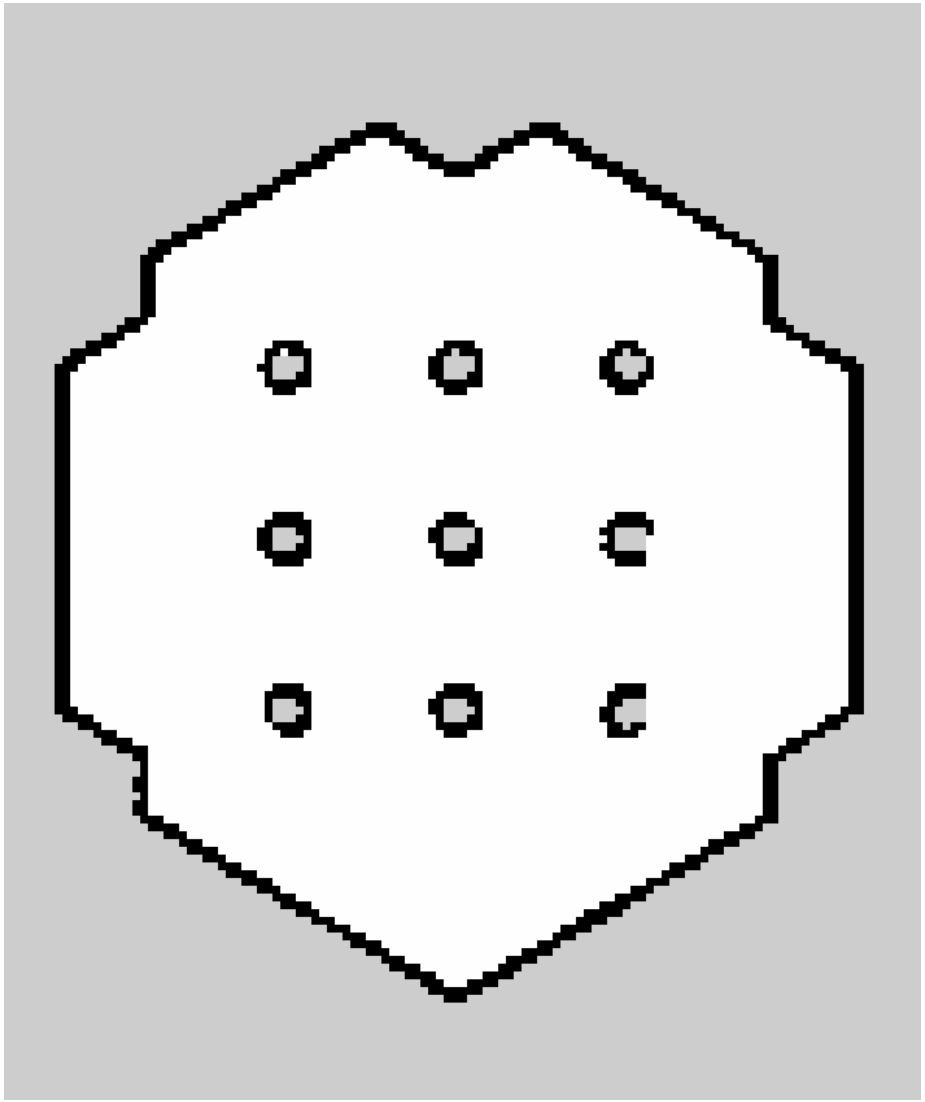
지도작성: Gmapping + TurtleBot3



https://youtu.be/7mEKrT_cKWI

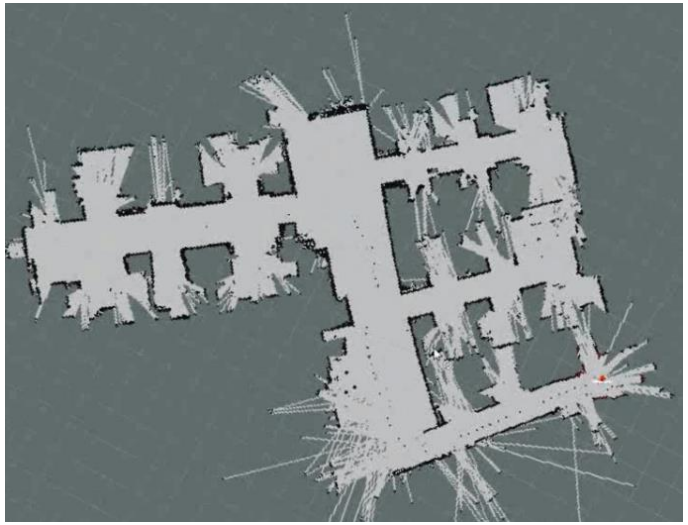
지도작성: Gmapping + TurtleBot3

- 완성된 지도

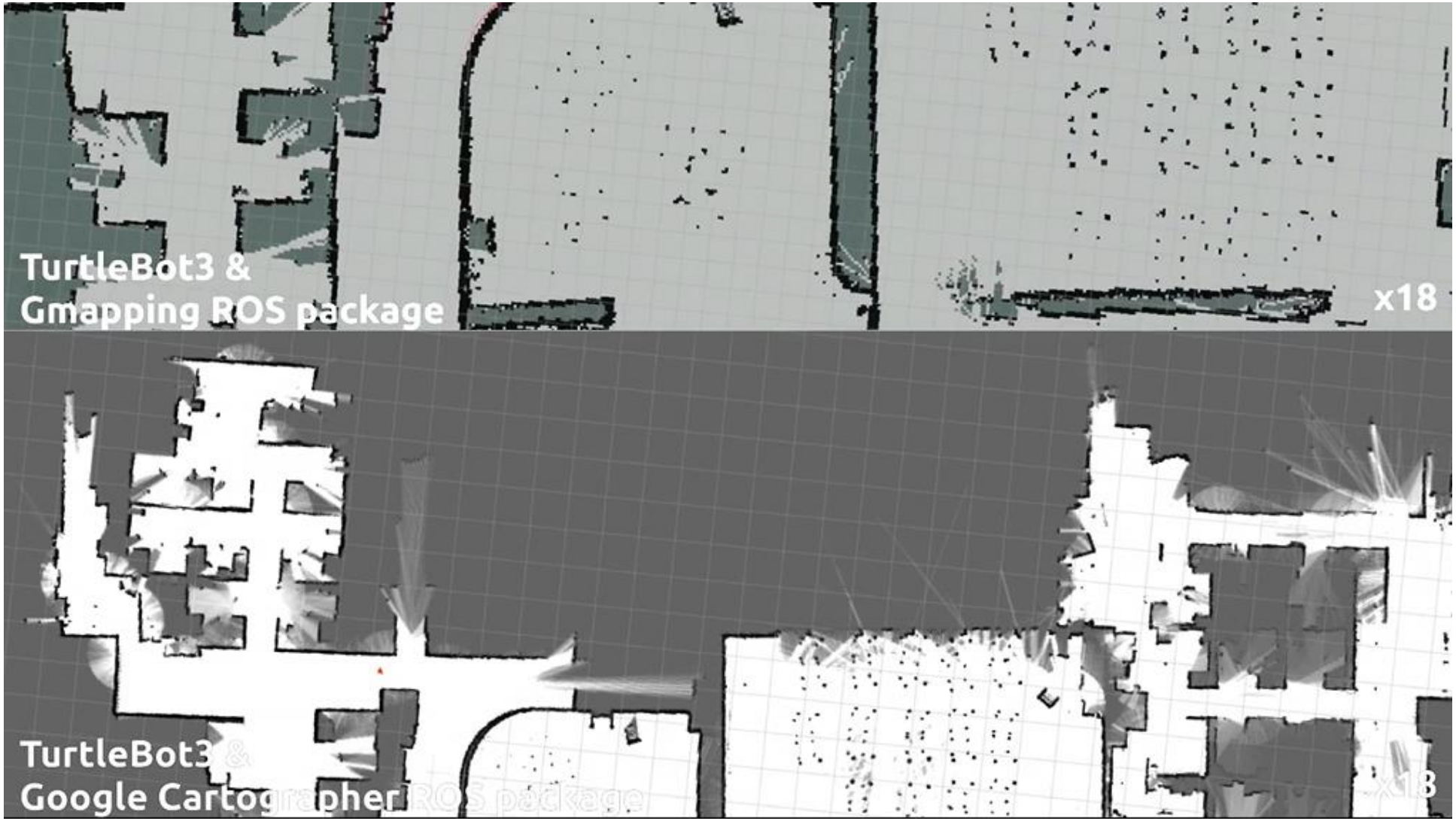


2차원 **점유 격자 지도**
(OGM, Occupancy Grid Map)

- 흰색 = 로봇이 이동 가능한 **자유 영역 (free area)**
- 흑색 = 로봇이 이동 불가능한 **점유 영역 (occupied area)**
- 회색 = 확인되지 않은 **미지 영역 (unknown area)**

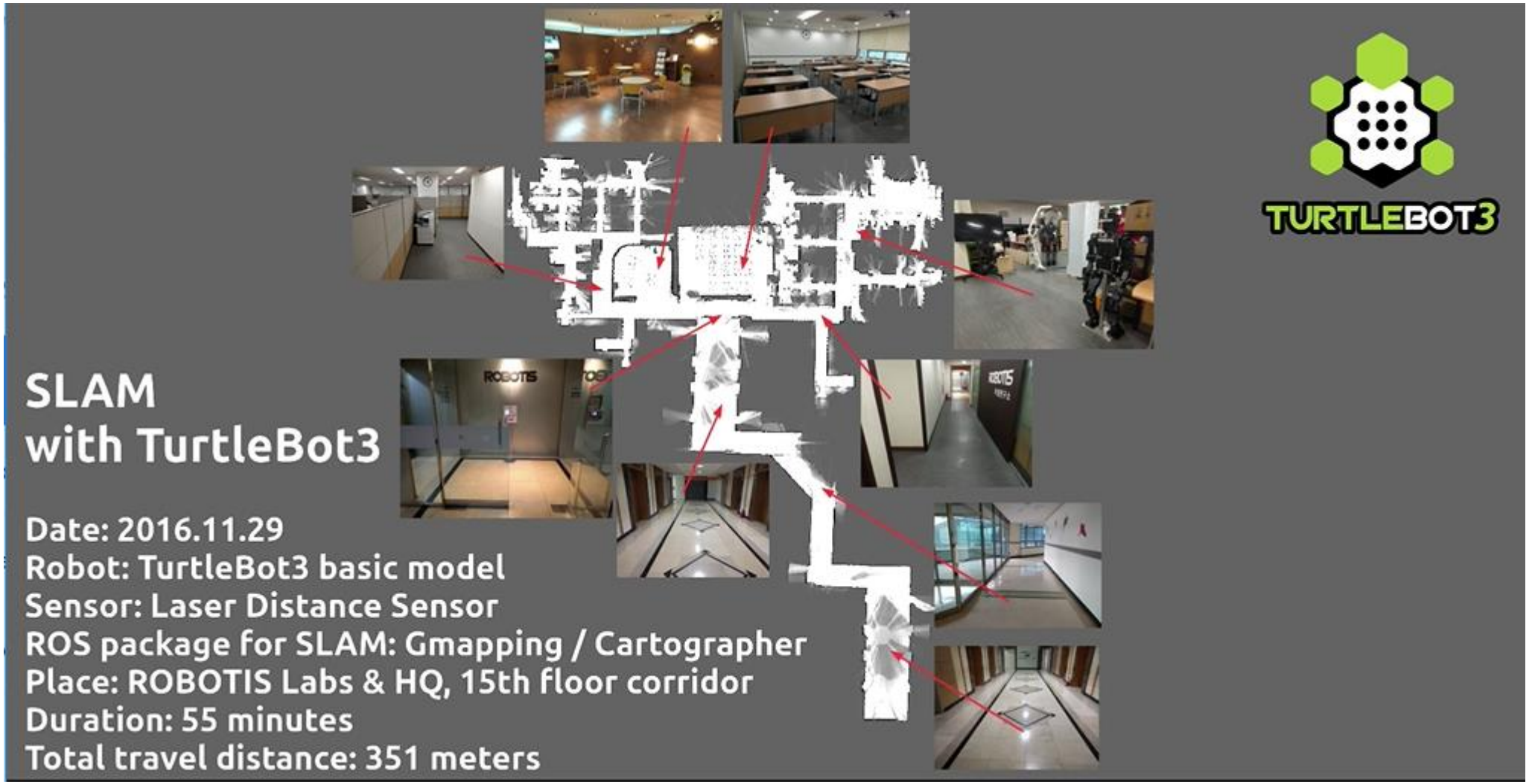


지도작성: Gmapping & Cartographer + TurtleBot3



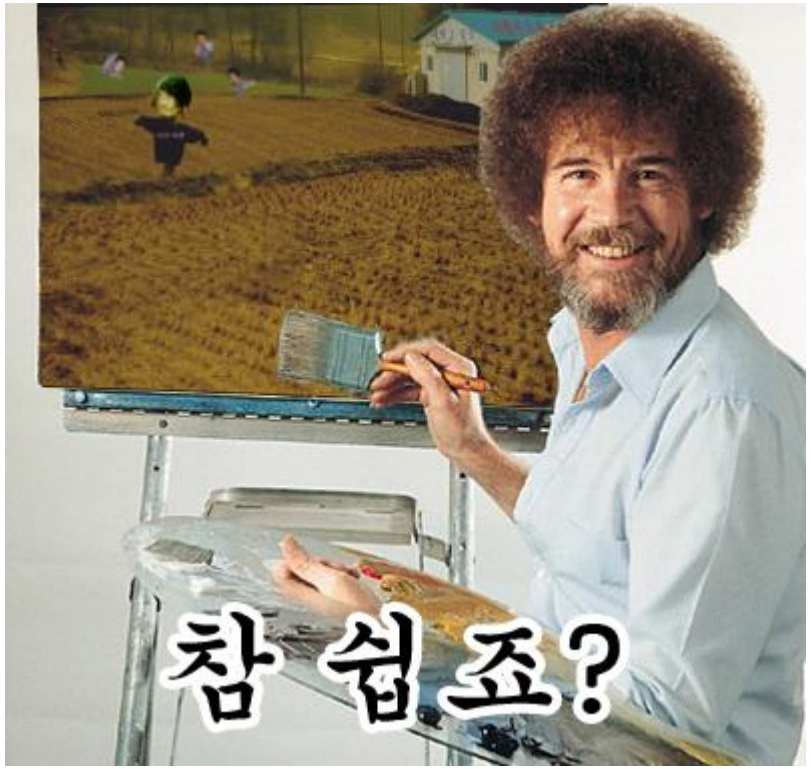
https://youtu.be/lkW4-dG2BCY

지도작성: Gmapping & Cartographer + TurtleBot3

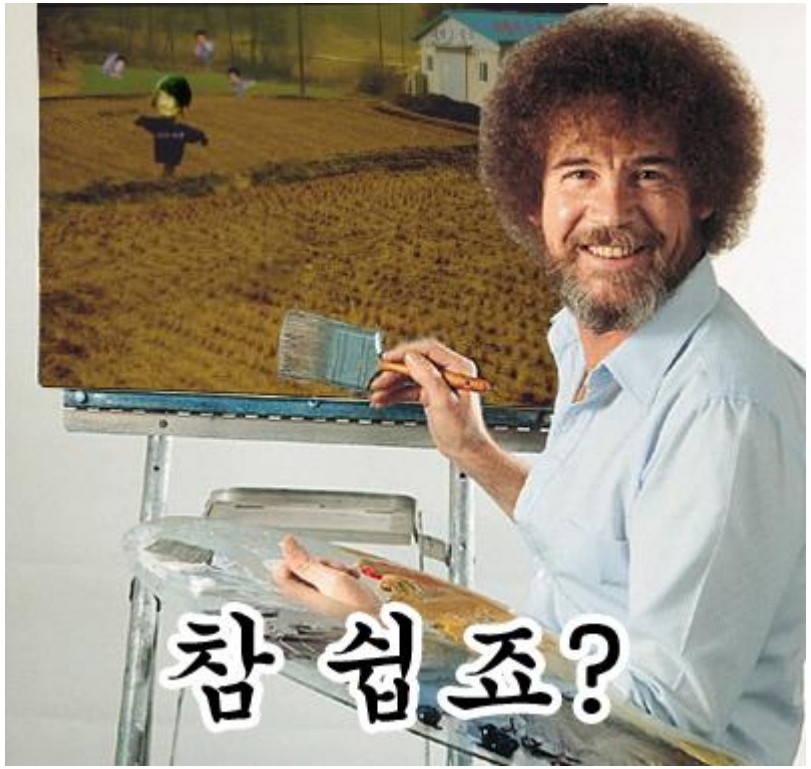


https://youtu.be/lkW4-dG2BCY

지도작성



지도작성



SLAM, Navigation 은 기본 기능이고
상위에 서비스 또는 모바일 로봇 자체를 하고 싶다고요?
그렇다면 SLAM, Navigation 은 그대로 쓰시고
좀 더 시간을 원하시는 부분에 투자하세요.
세상에 없는 유니크한 당신만의 로봇을 기대해 봅니다.

지도작성

SLAM, Navigation 은 기본 기능이고
상위에 서비스 또는 모바일 로봇 자체를 하고 싶다고요?
그렇다면 SLAM, Navigation 은 그대로 쓰시고
좀 더 시간을 원하시는 부분에 투자하세요.
세상에 없는 유니크한 당신만의 로봇을 기대해 봅니다.

SLAM, Navigation 을 더 공부하고 싶다고요?
모든 소프트웨어는 오픈 소스 입니다.
마음껏 보고, 이해해 보고, 기능도 추가하며
공부해 보세요. 이보다 더 좋은 교과서는 없습니다.

SLAM 관련 노드들의 처리 과정

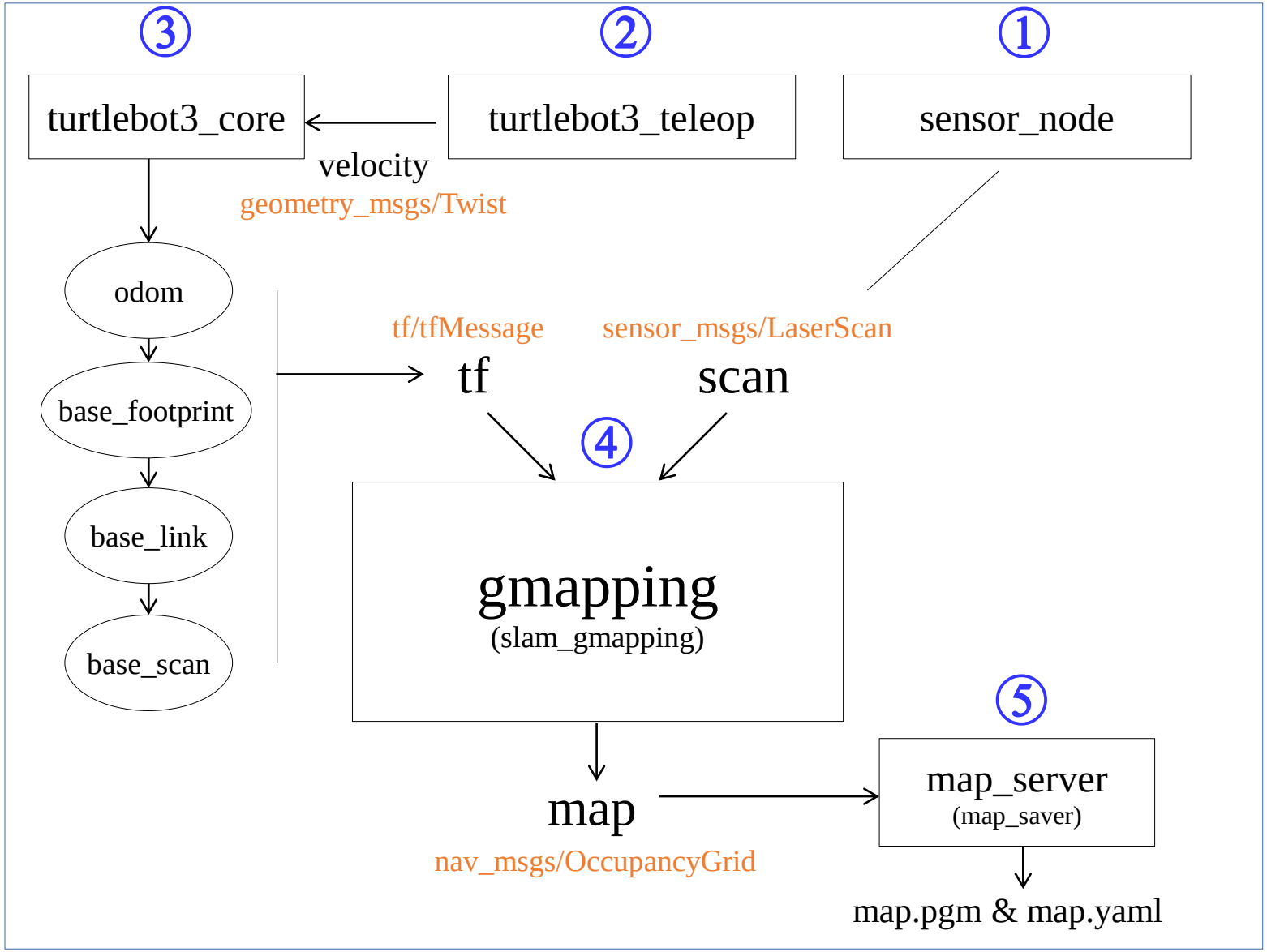
① sensor_node

② turtlebot3_teleop

③ turtlebot3_core

④ slam_gmapping

⑤ map_server

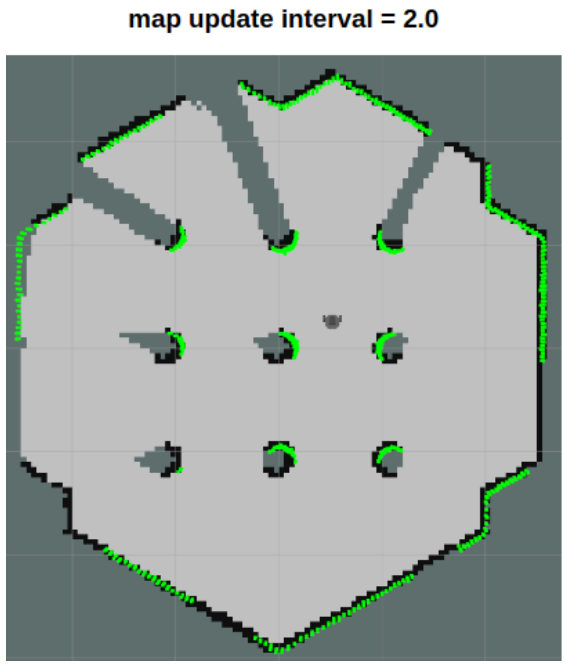


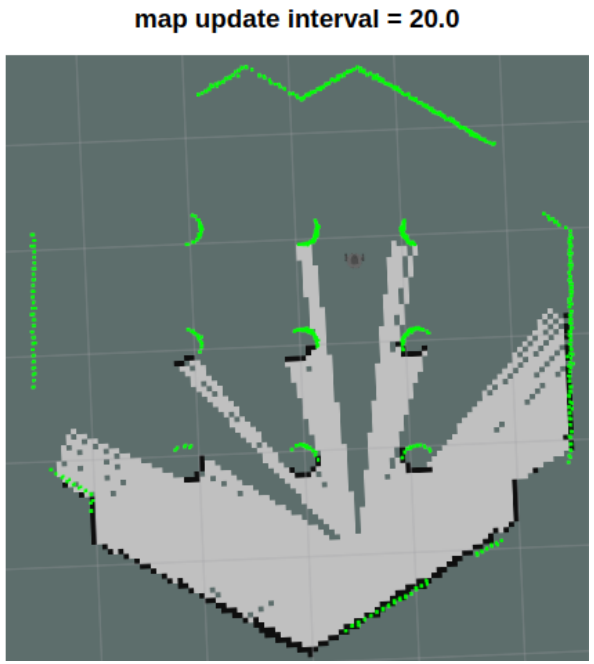
다양한 SLAM 적용하기

- Gmapping, Cartographer, Hector, Karto, Frontier Exploration

- SLAM Methods on TurtleBot3

- Tuning Guide



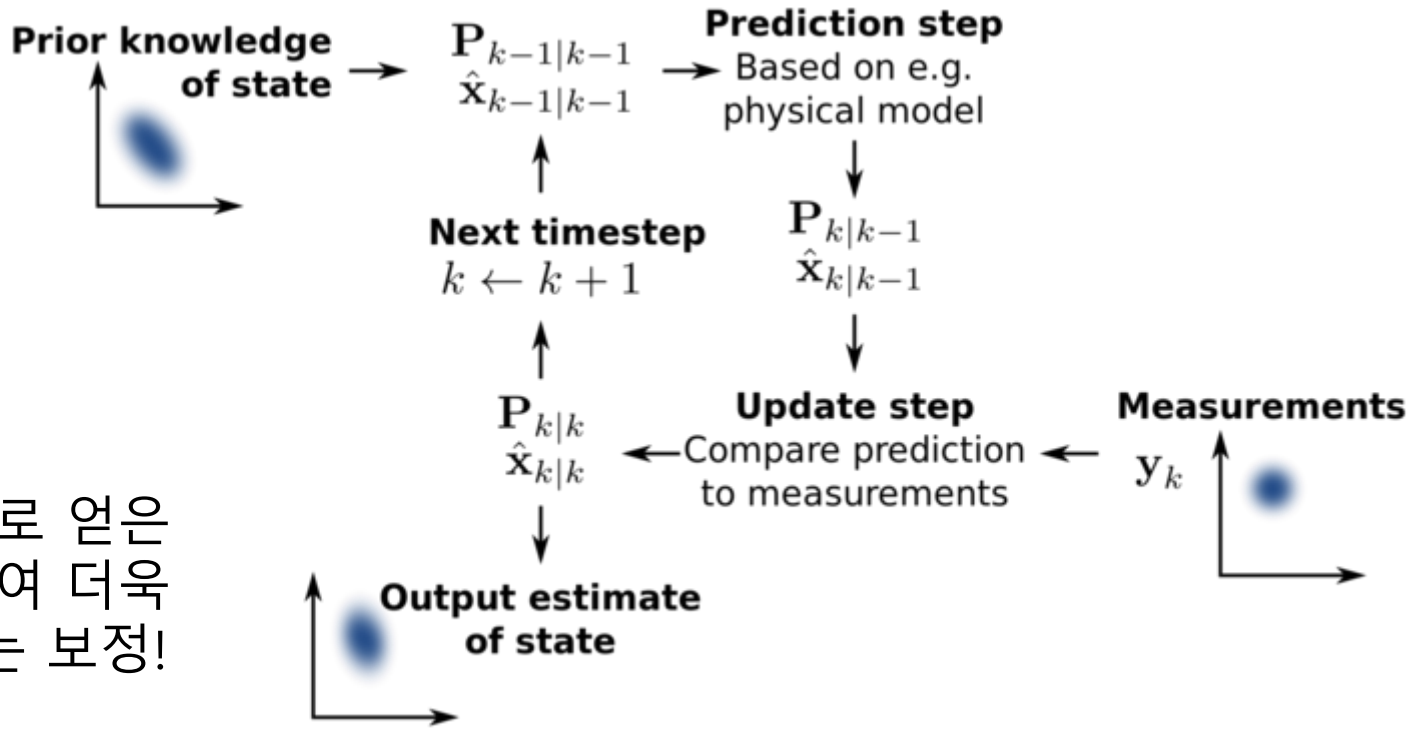


위치 추정(localization) | Kalman filter, Particle filter, Graph, Bundle adjustment

- **칼만 필터 (Kalman filter)**
  - 잡음이 포함되어 있는 선형 시스템에서 대상체의 상태를 추적하는 재귀 필터
  - 베이즈 확률 기반

  - 예측(Prediction)
    - 모델을 상정하고 이 모델을 이용하여 이전 상태로부터 현재 시점의 상태를 예측

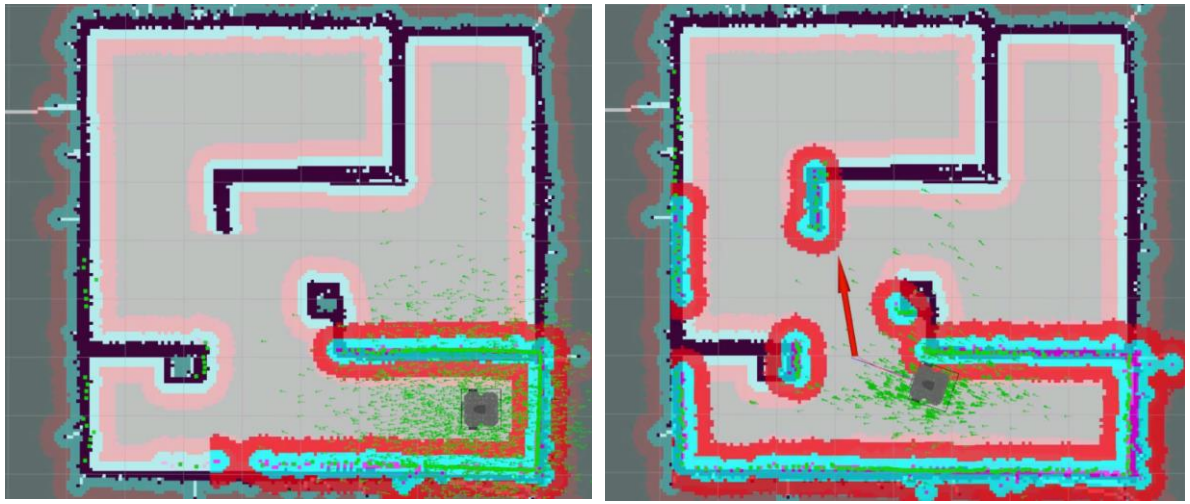
  - 보정(update)
    - 앞 단계의 예측 값과 외부 계측기로 얻은 실제 측정 값 간의 오차를 이용하여 더욱 정확한 상태의 상태 값을 추정하는 보정!



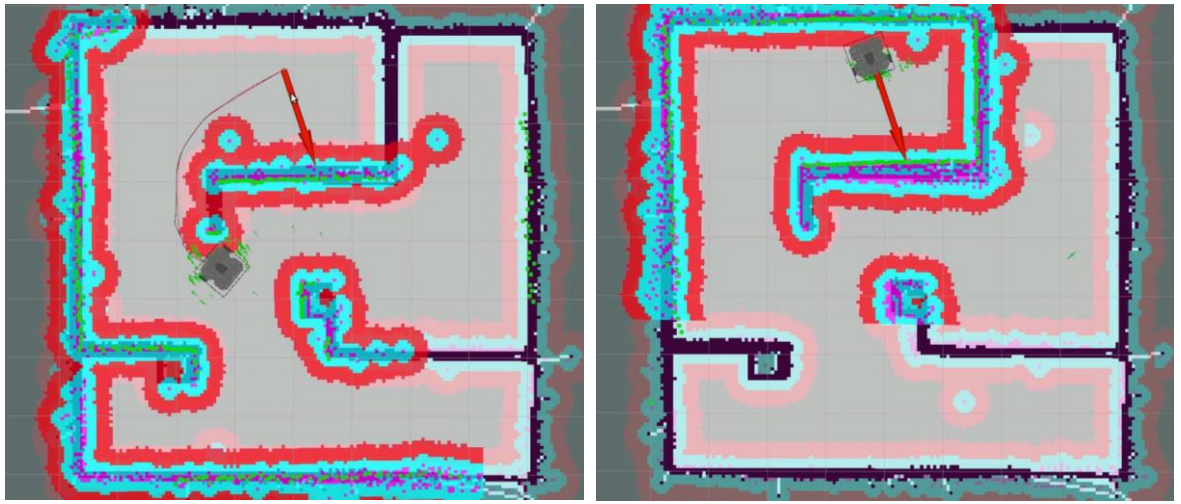
위치 추정(localization) | Kalman filter, Particle filter, Graph, Bundle adjustment

- 파티클 필터(Particle Filter)

- 파티클 필터는 시행 착오(try-and-error)법을 기반으로한 시뮬레이션을 통하여 예측하는 기술으로 대상 시스템에 확률 분포로 임의로 생성된 추정값을 파티클(입자) 형태로 나타낸다.



(t1) (t2)



(t3) (t4)

1) 초기화(initialization)
2) 예측(prediction)
3) 보정(update)
4) 위치 추정(pose estimation)
5) 재추출(Resampling)

S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics. MIT Press, Cambridge, MA, 2005.

내비게이션

내비게이션: Navigation + TurtleBot3

- http://emanual.robotis.com/docs/en/platform/turtlebot3/navigation/
- 마스터 실행 (Remote PC)

```
$ roscore
```

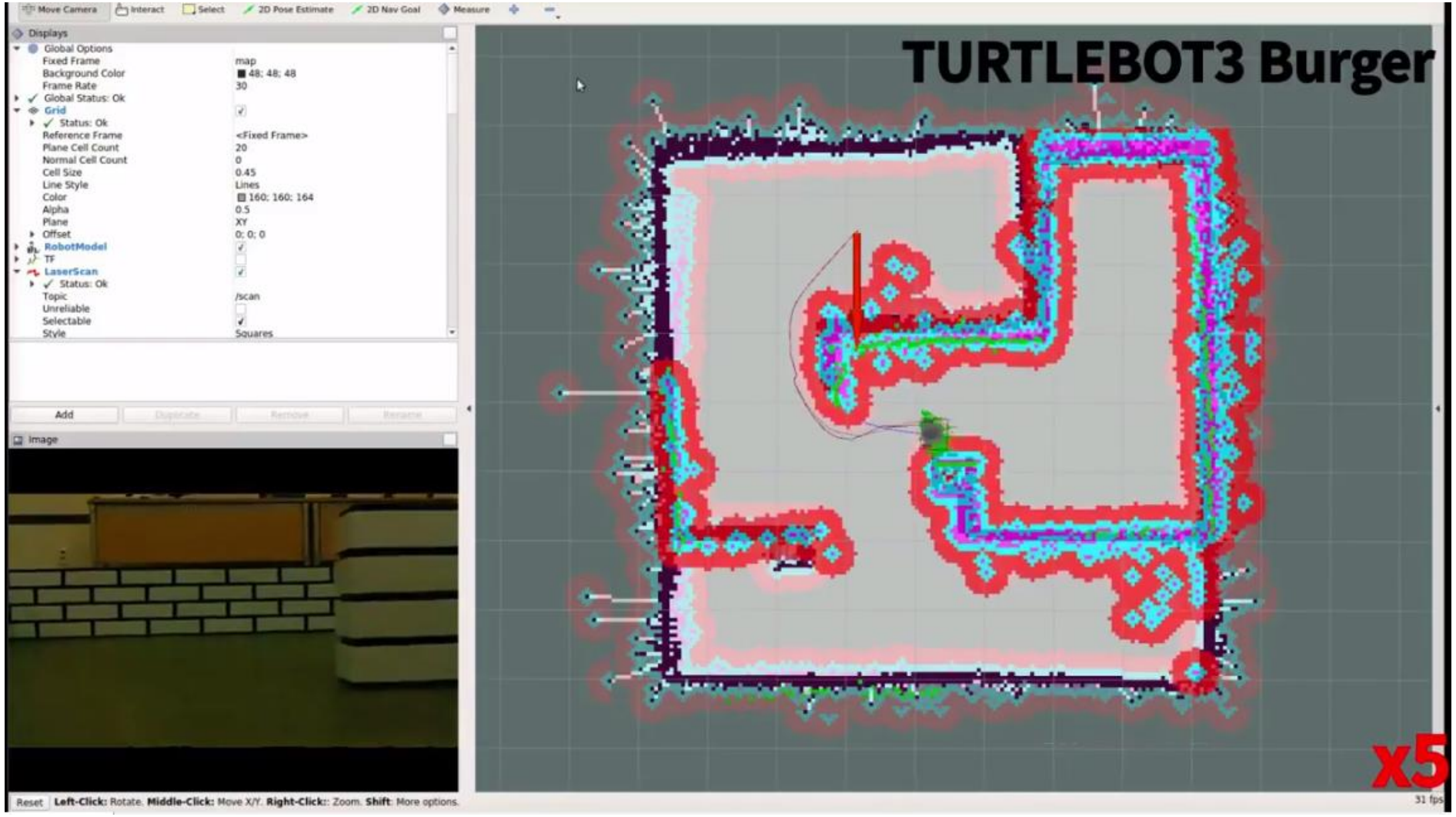
- 터틀봇 및 센서 구동 (SBC)

```
$ export TURTLEBOT3_MODEL=burger  (또는 waffle, waffle_pi)
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

- RViz, 터틀봇 원격 조종, 내비게이션 (Remote PC)

```
$ export TURTLEBOT3_MODEL=burger  (또는 waffle, waffle_pi)
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```

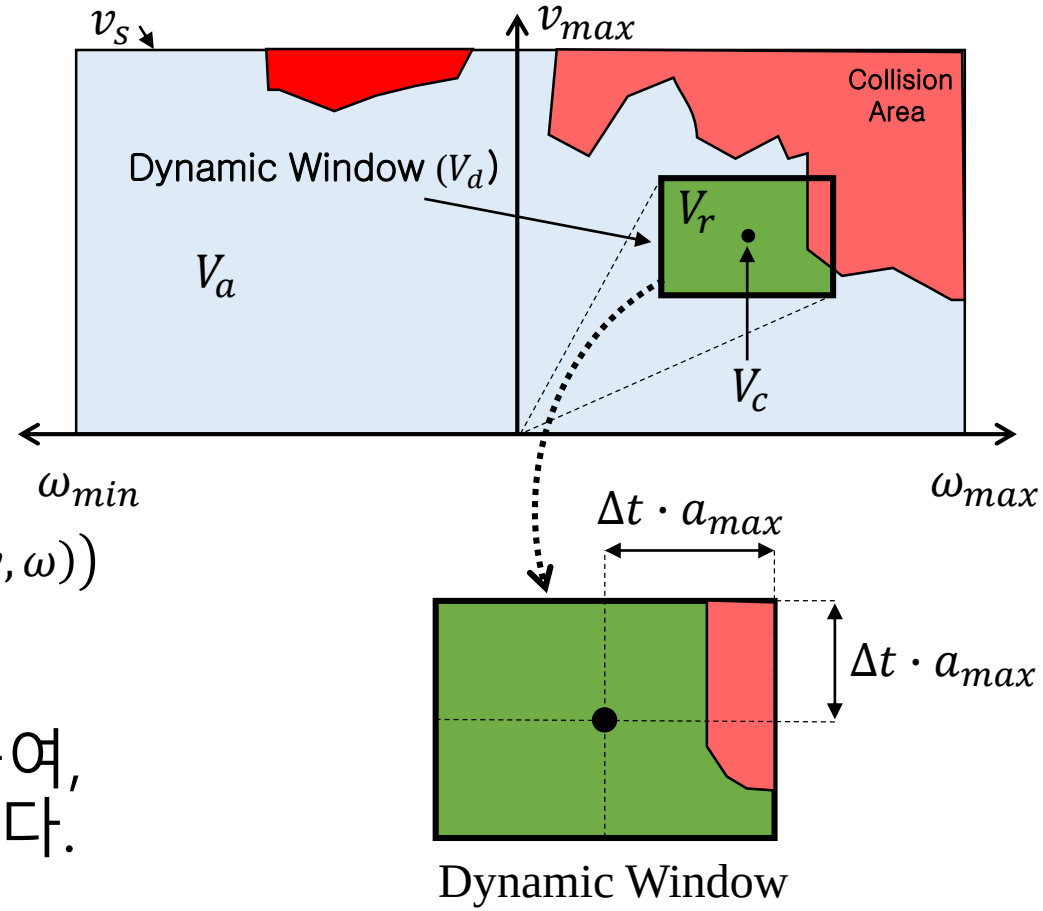
내비게이션



https://youtu.be/VYlMywwYALU

내비게이션

- **Dynamic Window Approach** (local plan에서 주로 사용)
- 로봇의 속도 탐색 영역(velocity search space)에서 로봇과 충돌 가능한 장애물을 회피하면서 목표점까지 빠르게 다다를 수 있는 속도를 선택하는 방법

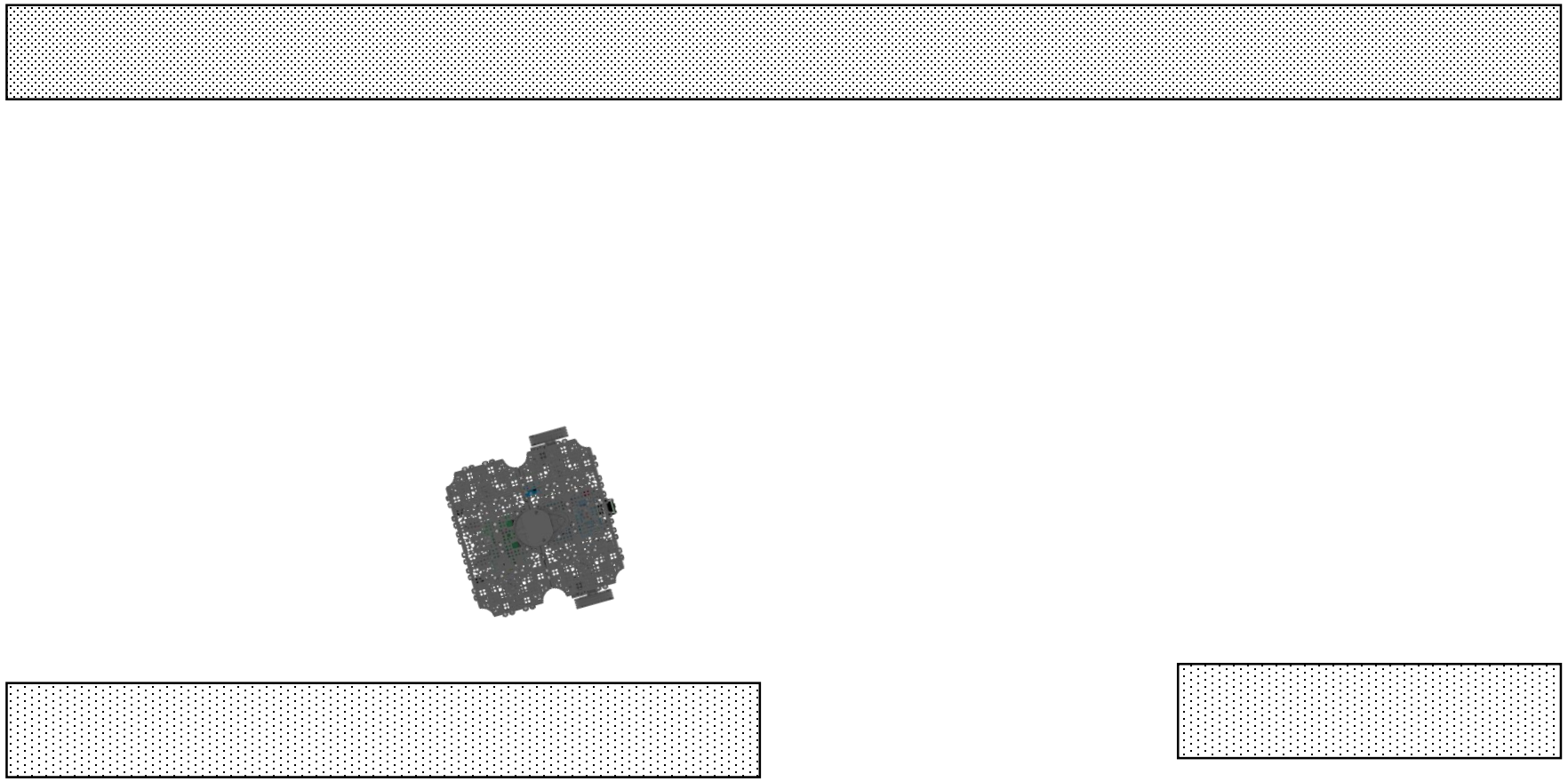


  - v (병진속도), ω (회전속도)
  - V_s: 가능 속도 영역
  - V_a: 허용 속도 영역
  - V_r: 다이나믹 윈도우 안의 속도 영역
  - $G_{(v,\omega)} = \sigma\big(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega)\big)$

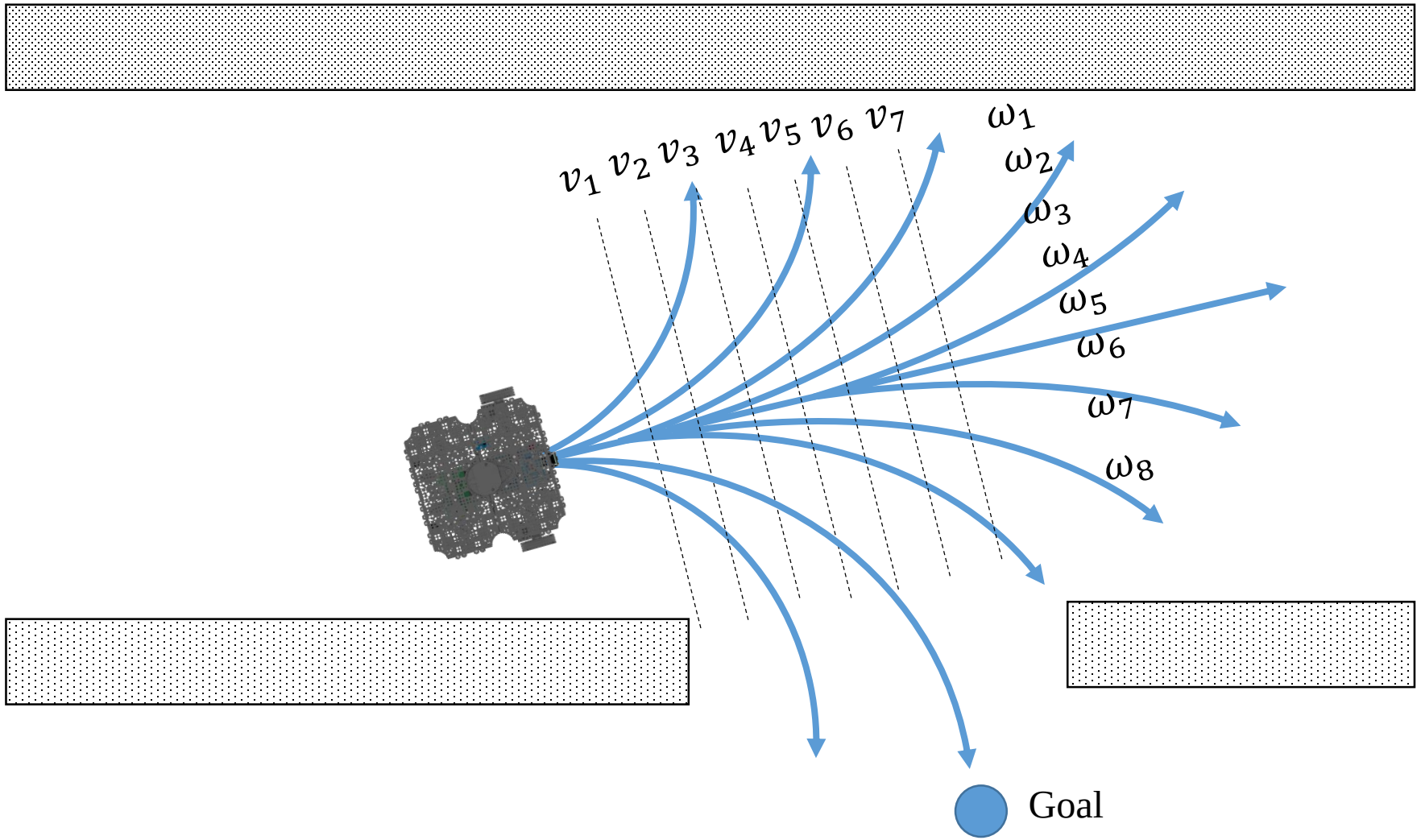
- 목적함수 G는 로봇의 방향, 속도, 충돌을 고려하여, 목적함수가 최대가 되는 속도 v, ω 를 구하게 된다.

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

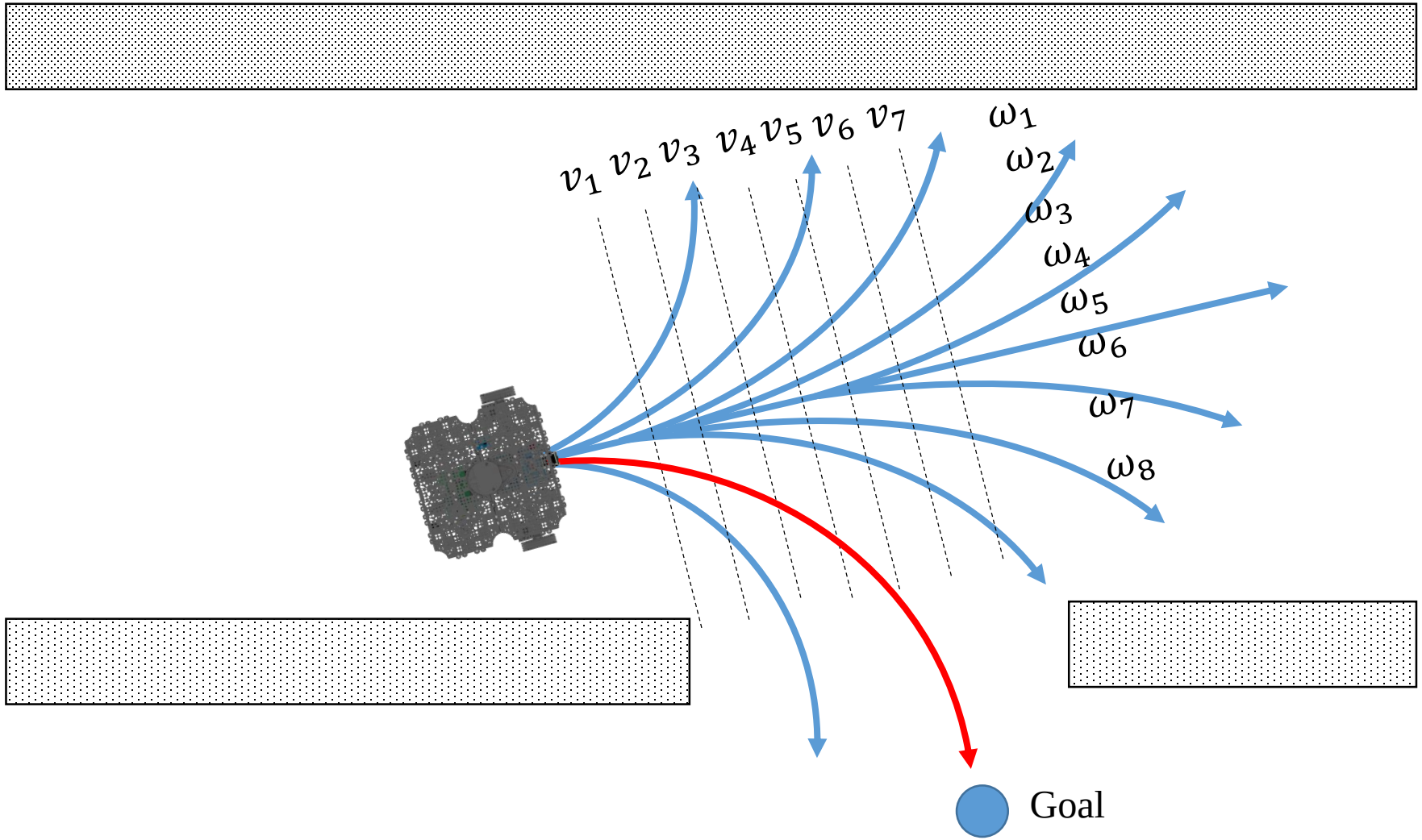
Dynamic Window Approach (DWA)



Dynamic Window Approach (DWA)



Dynamic Window Approach (DWA)



마지막으로... 다시 한번!

SLAM, Navigation 은 기본 기능이고
상위에 서비스 또는 모바일 로봇 자체를 하고 싶다고요?
그렇다면 SLAM, Navigation 은 그대로 쓰시고
좀 더 시간을 원하시는 부분에 투자하세요.
세상에 없는 유니크한 당신만의 로봇을 기대해 봅니다.

SLAM, Navigation 을 더 공부하고 싶다고요?
모든 소프트웨어는 오픈 소스 입니다.
마음껏 보고, 이해해 보고, 기능도 추가하며
공부해 보세요. 이보다 더 좋은 교과서는 없습니다.

실습 시간!

"SLAM / 내비게이션"

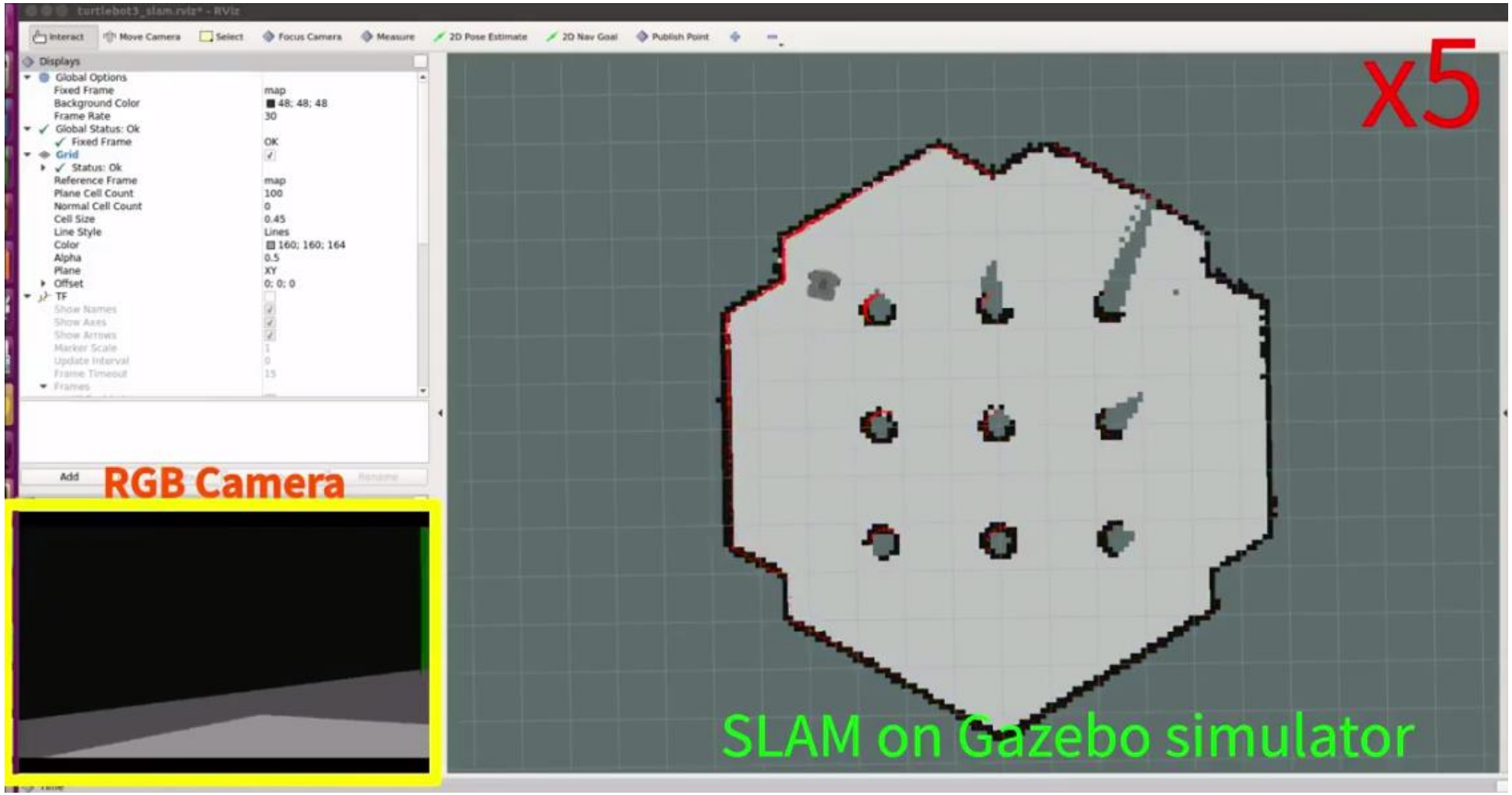
TurtleBot3 시뮬레이션 개발환경 준비

- 공식 터틀봇3 위키 참조
  - http://turtlebot3.robotis.com
- 기본 설치 패키지 (3차원 시뮬레이터 Gazebo를 이용하기 위한 준비)

```
$ sudo apt install ros-kinetic-joy ros-kinetic-teleop-twist-joy ros-kinetic-teleop-twist-keyboard ros-kinetic-laser-proc ros-kinetic-rgbd-launch ros-kinetic-depthimage-to-laserscan ros-kinetic-rosserial-arduino ros-kinetic-rosserial-python ros-kinetic-rosserial-server ros-kinetic-rosserial-client ros-kinetic-rosserial-msgs ros-kinetic-amcl ros-kinetic-map-server ros-kinetic-move-base ros-kinetic-urdf ros-kinetic-xacro ros-kinetic-compressed-image-transport ros-kinetic-rqt-image-view ros-kinetic-gmapping ros-kinetic-navigation
```

```
$ cd ~/catkin_ws/src/
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
$ cd ~/catkin_ws && catkin_make
```

TurtleBot3 in Gazebo



https://youtu.be/xXM5r_SVkWM

가상 로봇 실행 with Gazebo

- 3차원 시뮬레이터 Gazebo 상에서 가상의 로봇을 구동
  - Turtlebot3_teleop_key 노드를 통해 로봇 이동 가능
  - Rviz를 통해 Gazebo 상의 로봇에 장착된 센서 값 확인 가능
    - 2D Laser Range Sensor, Camera, Depth Camera, IMU 등

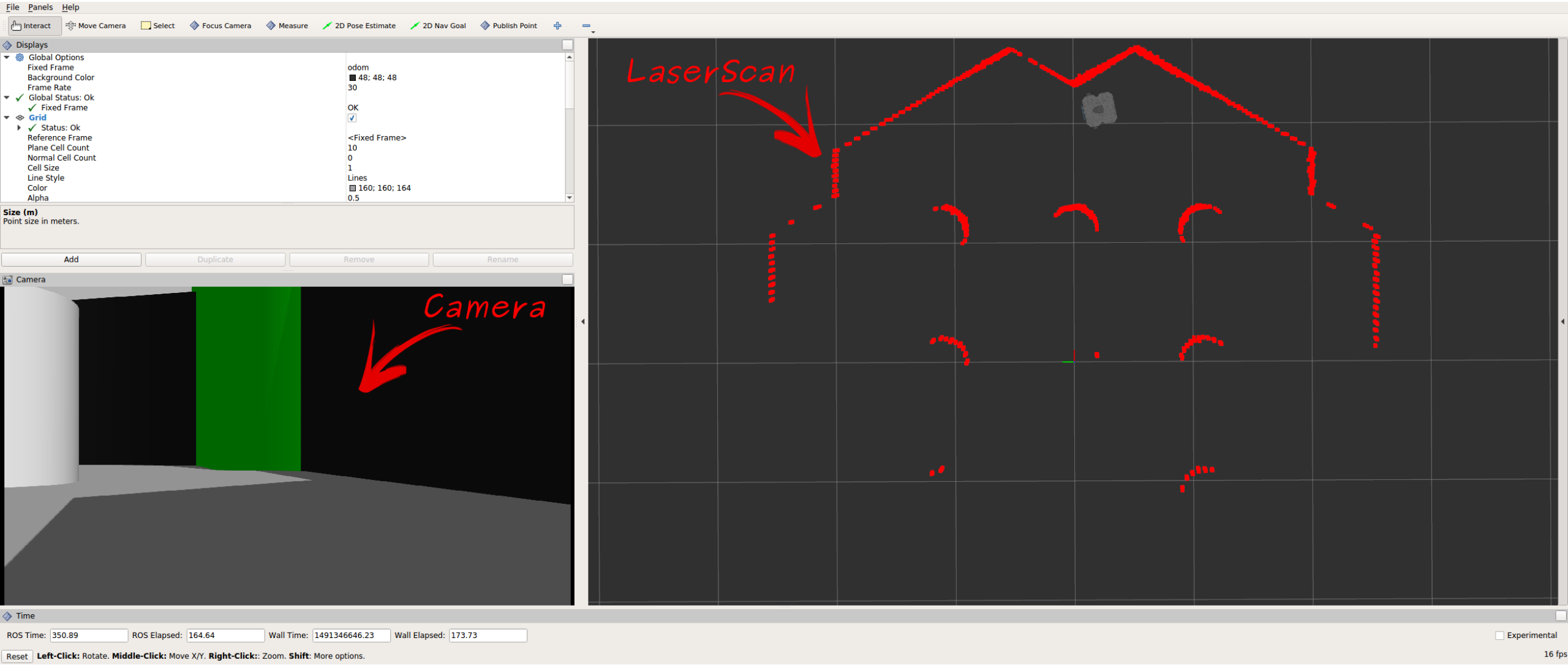
```
export TURTLEBOT3_MODEL=waffle_pi
```

(~/.bashrc 파일 맨 밑에 새로 추가해주세요.)

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

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

가상 로봇 실행 with Gazebo



가상 SLAM with Gazebo

```
export TURTLEBOT3_MODEL=waffle_pi
```

(~/.bashrc 파일 맨 밑에 새로 추가해주세요.)

- **Gazebo 실행**

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

- **SLAM 실행**

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

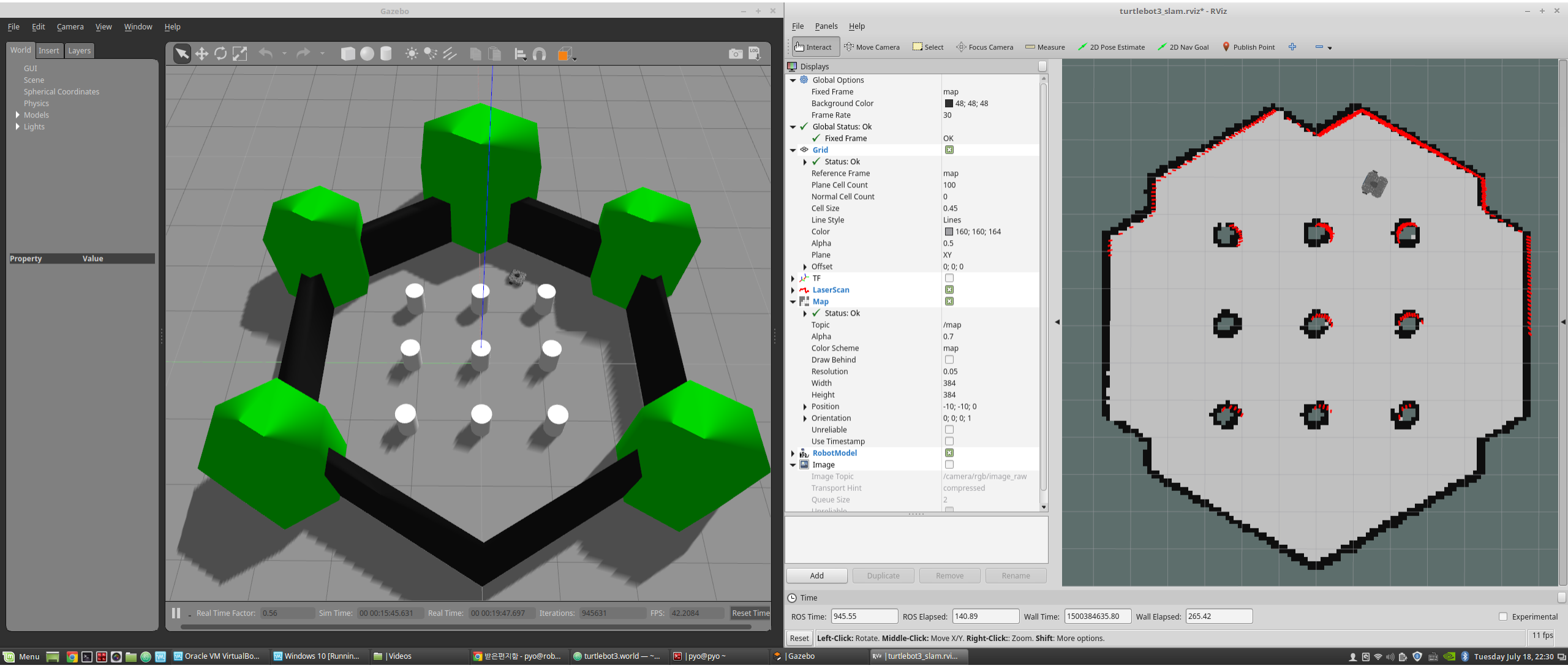
- **터틀봇 원격 조종**

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

- **지도 출력**

```
$ rosrun map_server map_saver -f ~/map
```

가상 SLAM with Gazebo



가상 내비게이션 with Gazebo

```
export TURTLEBOT3_MODEL=waffle_pi
```

(~/.bashrc 파일 맨 밑에 새로 추가해주세요.)

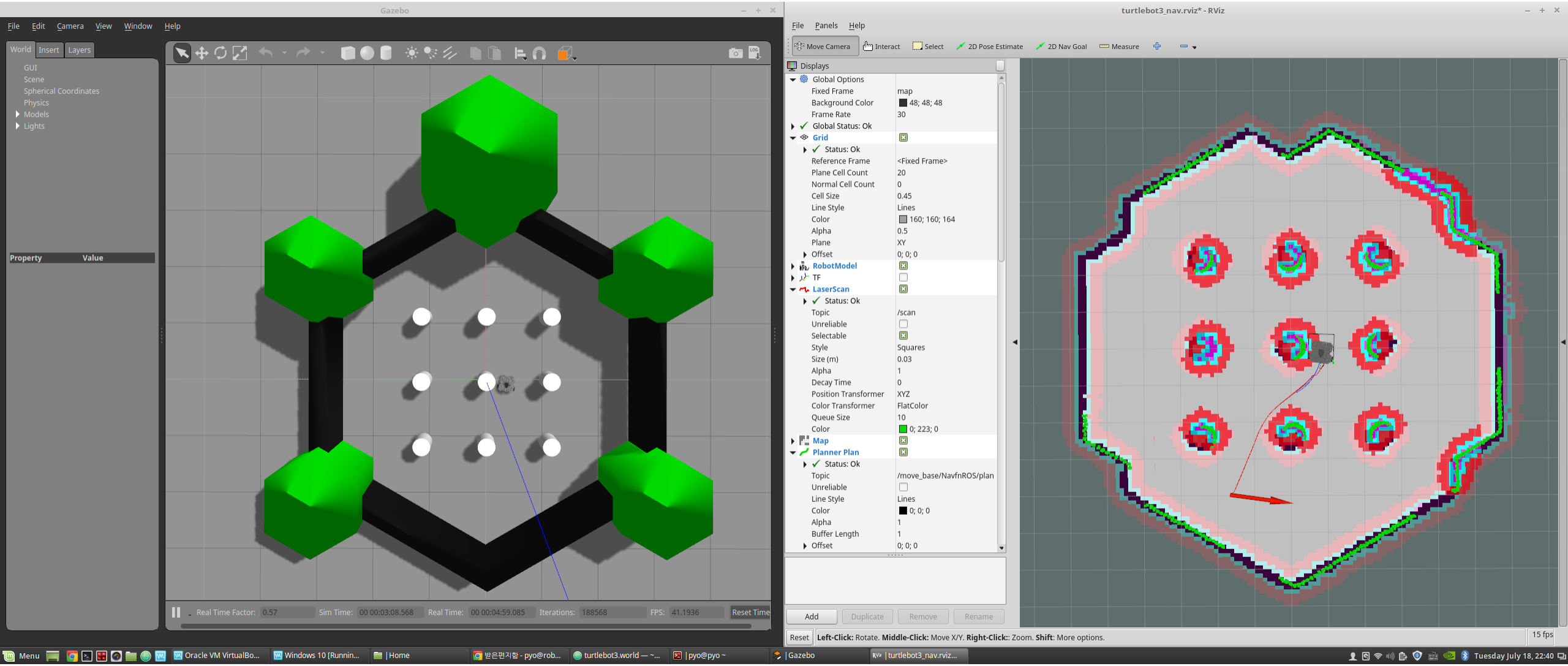
- **Gazebo 실행**

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

- **내비게이션 실행**

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

가상 내비게이션 with Gazebo



질문
대환영!

* 온라인 상의 질문이라면
오르카 및 로열로를 이용해주세요!

여기서! 광고 하나 나가요~

개정 증보판

✔ 한국어판 구매 링크

✔ 4개 언어로 출판!

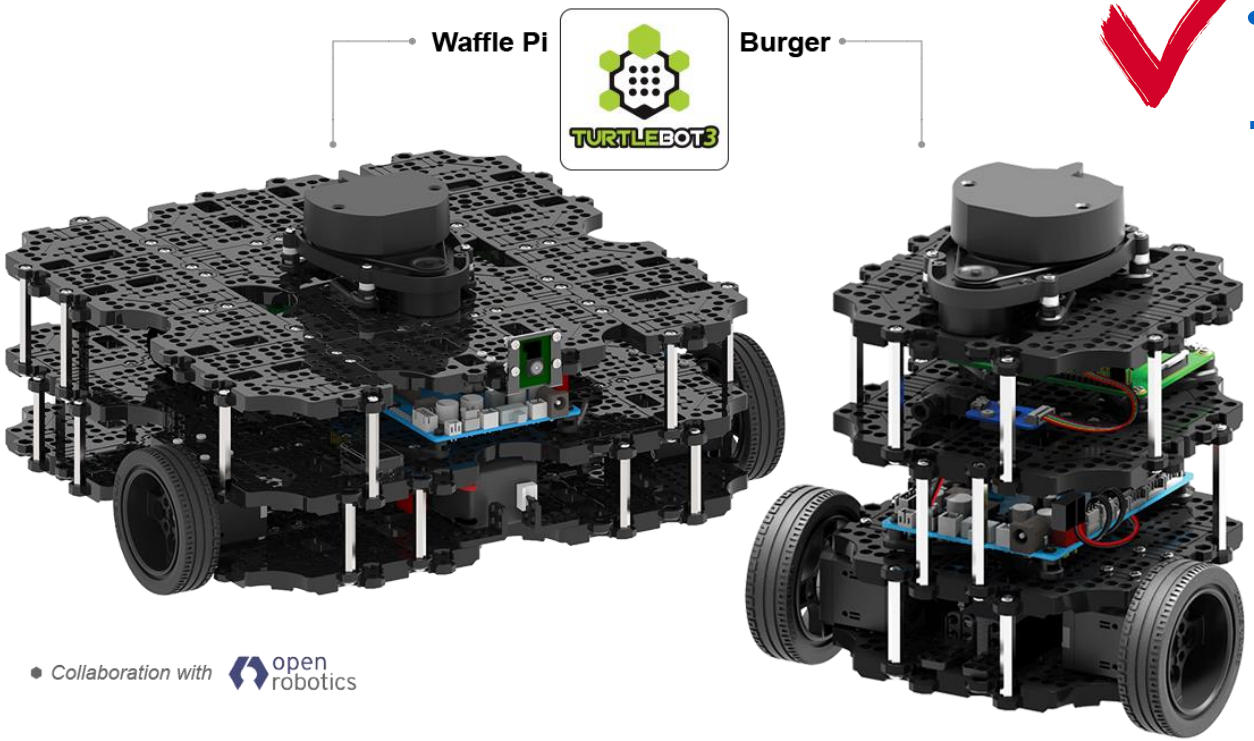
국내 유일! 최초! ROS 참고서!
ROS 공식 플랫폼 **TurtleBot3** 개발팀이
직접 저술한 바이블급 ROS 책

여기서! 광고 둘 나가요~

TURTLEBOT3

인공지능(AI) 연구의 시작,
ROS 교육용 공식 로봇 플랫폼

터틀봇3는 ROS기반의 저가형 모바일 로봇으로 교육, 연구, 제품개발, 취미 등 다양한 분야에서 활용할 수 있습니다.



Direct Link

여기서! 광고 셋 나가요~

오로카

- 오로카
- www.oroca.org
- 오픈 로보틱스 지향
- 풀뿌리 로봇공학의 저변 활성화
- 공개 강좌, 세미나, 프로젝트 진행

- 로봇공학을 위한 열린 모임 (KOS-ROBOT)
- www.facebook.com/groups/KoreanRobotics
- 로봇공학 통합 커뮤니티 지향
- 일반인과 전문가가 어울러지는 한마당
- 로봇공학 정보 공유
- 연구자 간의 협력

Robot Source

- RobotSource
- www.robotsource.org
- 글로벌 로보틱스 커뮤니티 지향
- 로봇공학 정보 공유
- 자신의 로봇 프로젝트 공유
- DIY 로봇 프로젝트 진행

혼자 하기에 답답하시다고요?

커뮤니티에서 함께 해요~

끝.

표윤석

Yoonseok Pyo
pyo@robotis.com
www.robotpilot.net

www.facebook.com/yoonseok.pyo