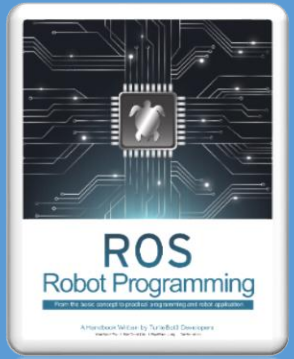
SLAM & Navigation



Textbook
P. 312~360

Contents

You Tube Subscribe

Textbook
P. 312~360

Let's start !

This topic is

SLAM, Navigation!

Simultaneous Localization And Mapping
&
Navigation

What?

——;;

Let’s go a little easier!

Path Finding

How is it?

Path...

Path 「Noun」

1. A way beaten, formed, or trodden by the feet persons or animals
2. A narrow walk or way
3. A route, course, or track along which something moves

Path Finding...

Path 「Noun」

1. A way beaten, formed, or trodden by the feet persons or animals
2. A narrow walk or way
3. A route, course, or track along which something moves

Long companion of travel

compass & map

What if there is
no compass & map?

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

I’m here.

- Travelers with the Sun, the Moon, and stars
- Compass, one of Four great Chinese inventions
- Development of Compass
  - Magnetic compass
  - Front compass
  - GPS
- Map

Big Dipper, by Magnus Manske, Public Domain

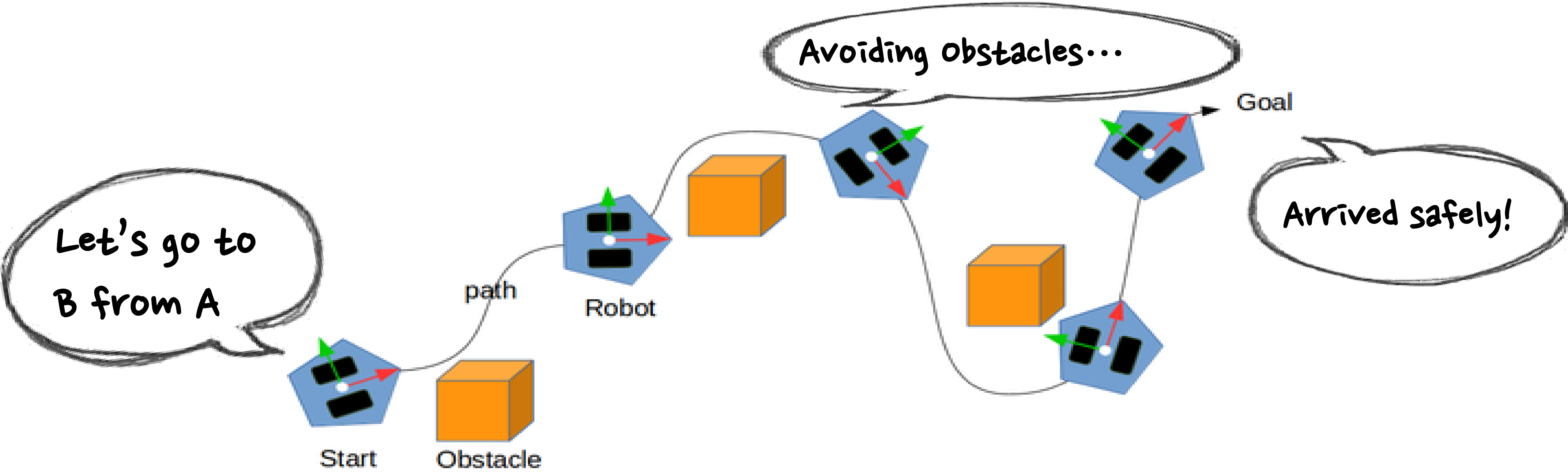
pixabay.com, CC0

Imagine it!
Find Path in the dark

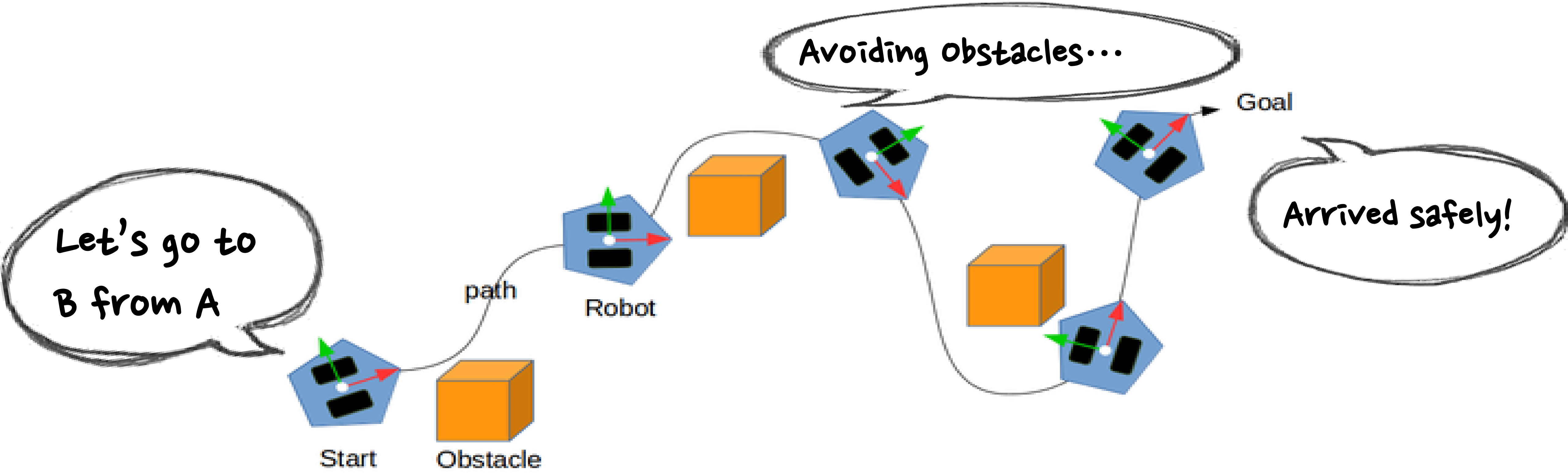
Path Finding of Robot

(From now on,
Let's learn more in detail)

What you need for path finding!



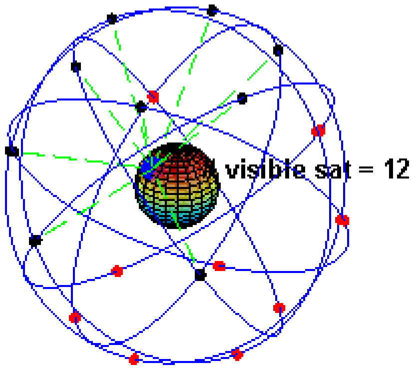
What you need for path finding!



① **Position**: Measuring/estimating the robot's position
② **Sensing**: Measuring obstacles such as walls and objects
③ **Map**: Maps with road and obstacle information
④ **Path**: Calculate optimal path to the destination and follow the path

① Position: Measuring/estimating the robot's position

- GPS (Global Positioning System)



  - Error
  - Weather
  - Outdoor

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

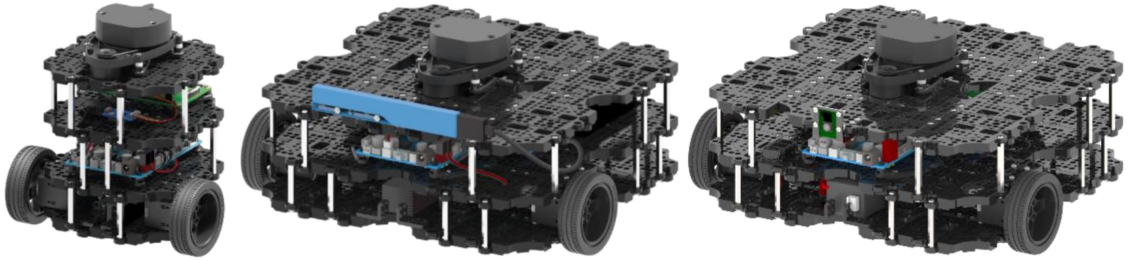
Estimote (Beacon)

StarGazer

Vicon MX

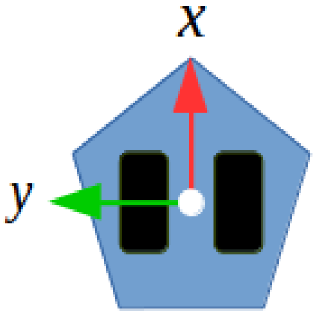
① Position: Measuring/estimating the robot's position

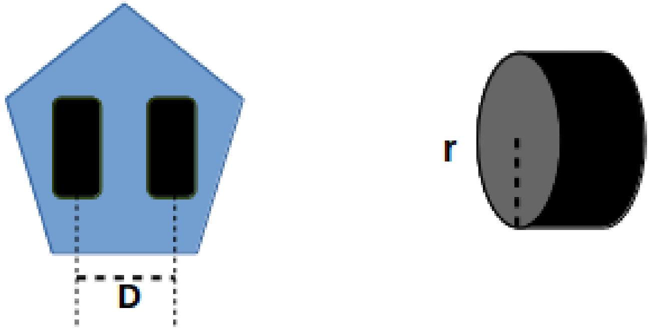
- Dead Reckoning
  - Using the encoder value of both wheel axes
  - Calculate moving distance and rotation angle, and then estimate position
  - Floor slip, mechanical, cumulative error
  - Position compensation with inertial sensor, filter such as IMU
  - Kalman filter...



TurtleBot 3

- Required Information
  - Encoder value E on both wheel axes (Recalculated as gear ratio for motor shaft)
  - Distance between wheels D
  - Wheel radius r

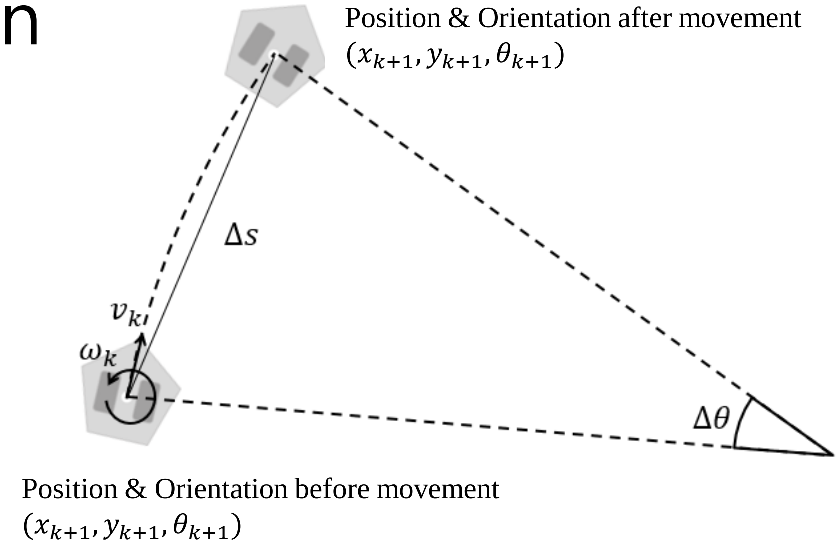


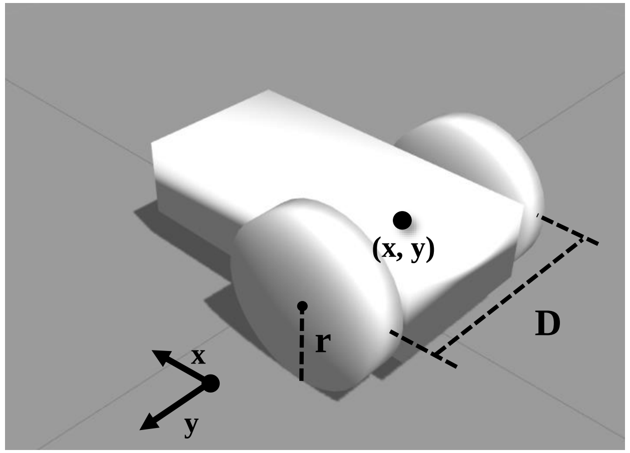


① Position: Measuring/estimating the robot's position

- Dead Reckoning Calculation
  - Linear velocity: v
  - Angular velocity: w

- Use Runge-Kutta Formula
  - Approximate value of moved position x, y
  - Rotation angle θ





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

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

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

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

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

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

$$\Delta \mathrm{s} = v_k T_e \qquad \Delta\theta = \omega_k T_e$$

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

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

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

② Sensing: Measuring obstacles such as walls and objects

- Distance Sensor
  - LRF, ultrasonic sensor, infrared distance sensor

- Vision Sensor
  - Stereo camera, mono camera, omni-directional camera

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

③ Map: Maps with road and obstacle information

- Robots need a map to find a path!
- Map
  - Digital maps for infrastructure such as roads!
  - Maps of hospitals, cafes, companies, homes?
  - Maps of unknown, collapsed hazardous areas?

pixabay.com, CC0

③ Map: Maps with road and obstacle information

- Robots need a map to find a path!
- Map
  - Digital maps for infrastructure such as roads!
  - Maps of hospitals, cafes, companies, homes?
  - Maps of unknown, collapsed hazardous areas?
- **Map? Let's make it!**
- **SLAM**

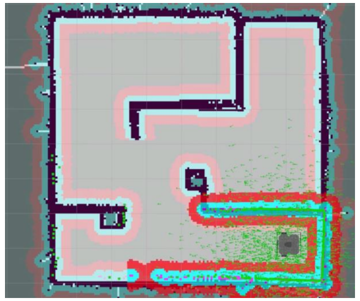
  (Simultaneous Localization And Mapping)

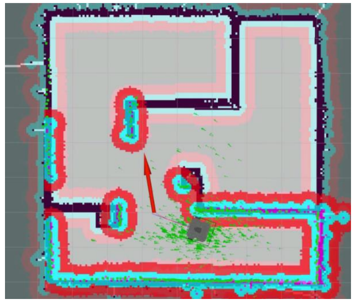
  Together **Where am I?** **Build a map**

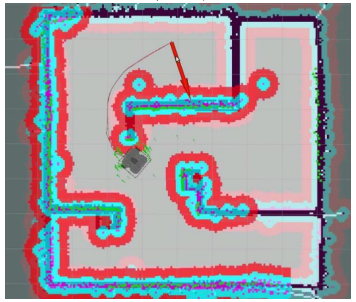
④ Path: Function to calculate optimal path to destination and travel

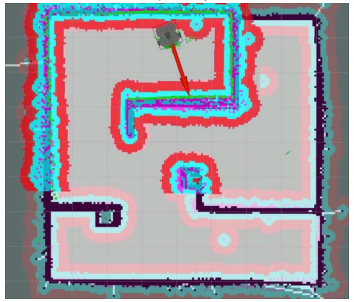
- Navigation
- Localization / Pose estimation
- Path search and planning

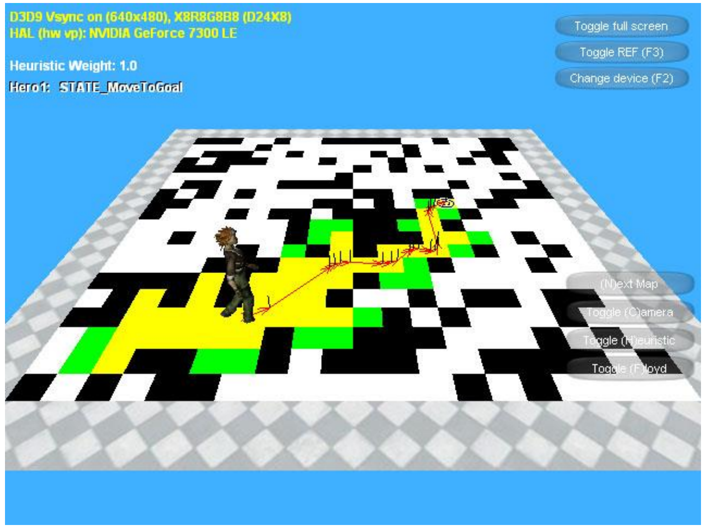
- Dynamic Window Approach (DWA)
- A* algorithm (A Star)
- Potential Field
- Particle Filter
- Graph

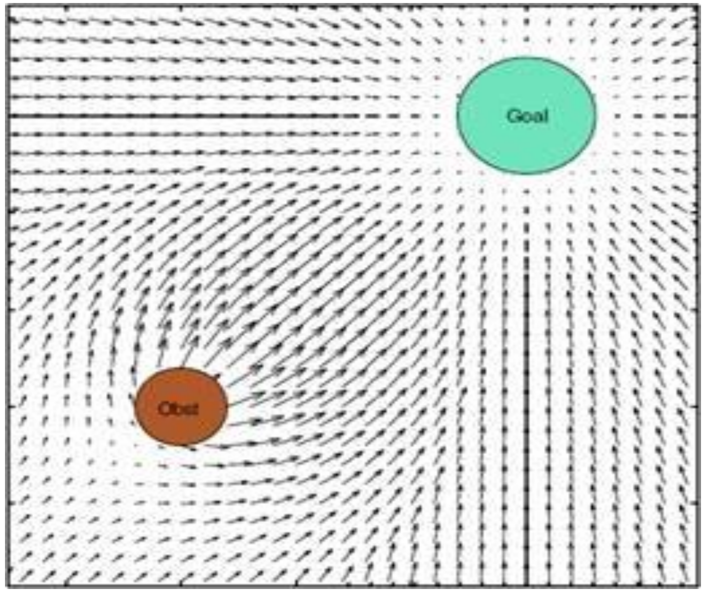

(t1)


(t2)


(t3)


(t4)





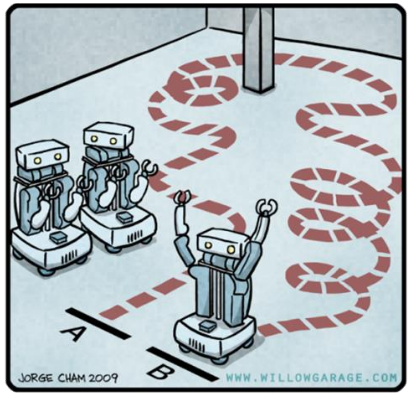
https://students.cs.byu.edu/~cs470ta, http://vimeo.com/3423169

① **Position**

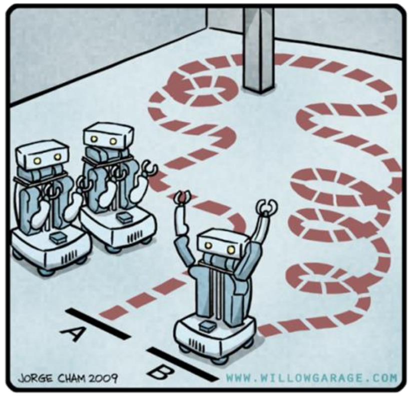
② **Sensing**

③ **Map**

④ **Path**



① **Position** ② **Sensing** ③ **Map** ④ **Path**



Position+Sensing → **Map**

SLAM

Position+Sensing+Map → **Path**

Navigation

SLAM

Gmapping

- One of SLAM method published in OpenSLAM, packaged in ROS
- Author: G. Grisetti, C. Stachniss, W. Burgard
- Feature: Rao-Blackwellized particle filter, Decreased number of particles, grid map

- Hardware Constraints
  - **X, Y, Theta Speed Command**
    - **Differential drive mobile robot**
    - **Omni-wheel robot**
  - **Odometry**
  - **Measuring sensor: 2D plane measurable sensor(LRF, LiDAR, Kinect, Xtion, etc.)**
  - Rectangular and circular robots

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

Mapping: Gmapping + TurtleBot3

- Software Preparation
  - 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 Packages
  - 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

Mapping: Gmapping + TurtleBot3

- http://turtlebot3.robotis.com/en/latest/slam.html
- Run Master (Remote PC)

```
$ roscore
```

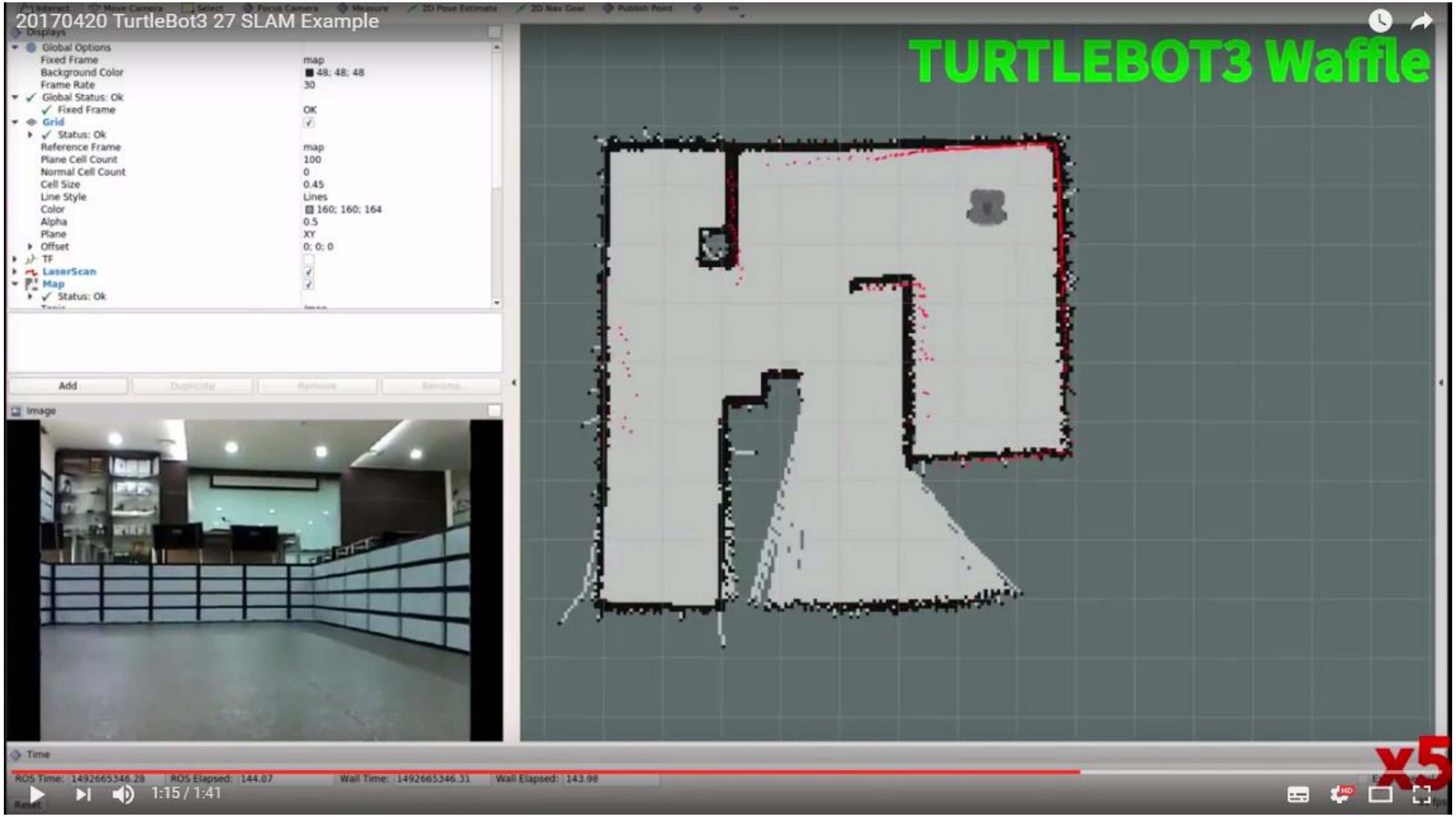
- TurtleBot3 and Sensor Drive (SBC)

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

- RViz, TurtleBot3 Remote Control, Mapping (Remote PC)

```
$ export TURTLEBOT3_MODEL=burger  (or waffle or waffle_pi)
$ roslaunch turtlebot3_slam turtlebot3_slam.launch
$ rosrun rviz rviz -d `rospack find turtlebot3_slam`/rviz/turtlebot3_slam.rviz
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
$ rosrun map_server map_saver -f ~/map
```

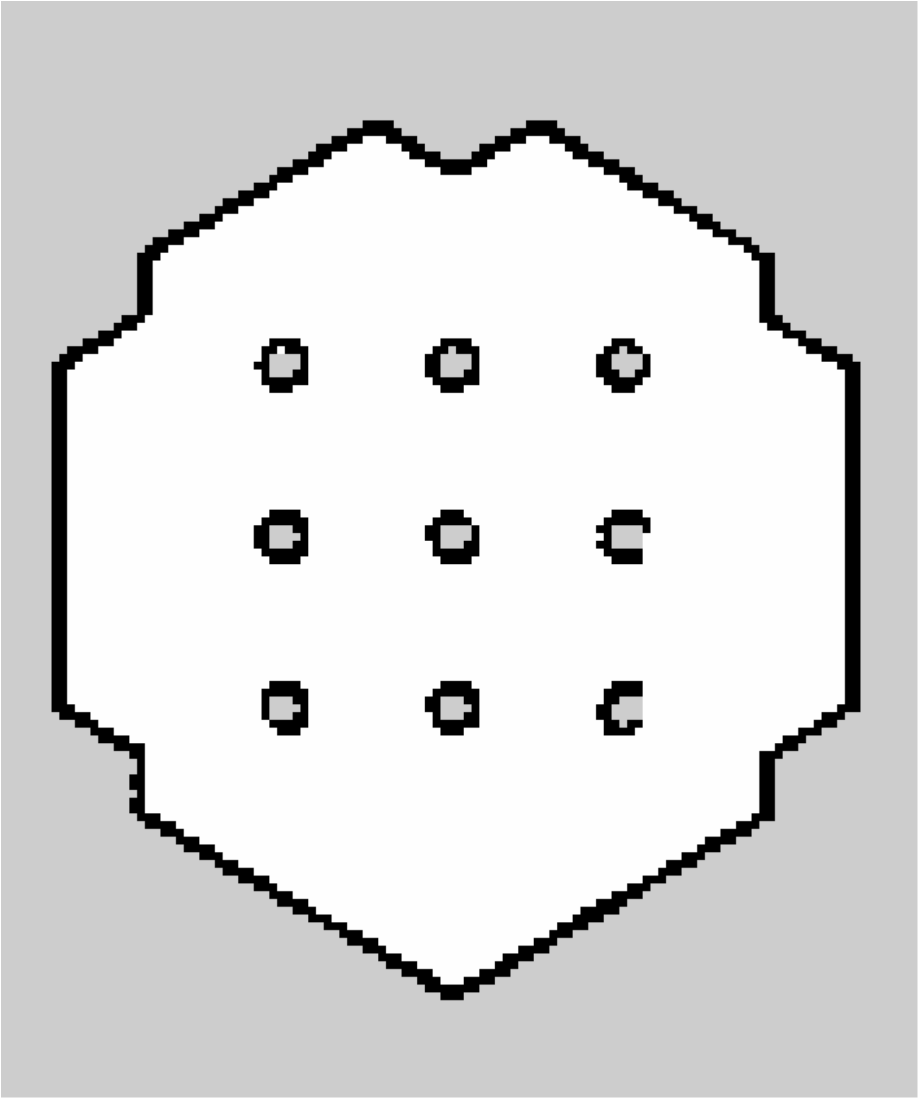
Mapping: Gmapping + TurtleBot3



https://youtu.be/7mEKrT_cKWI

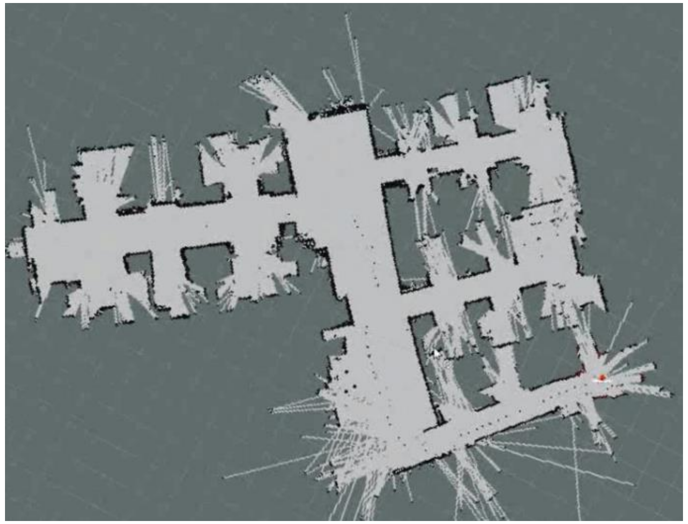
Mapping: Gmapping + TurtleBot3

- Completed map

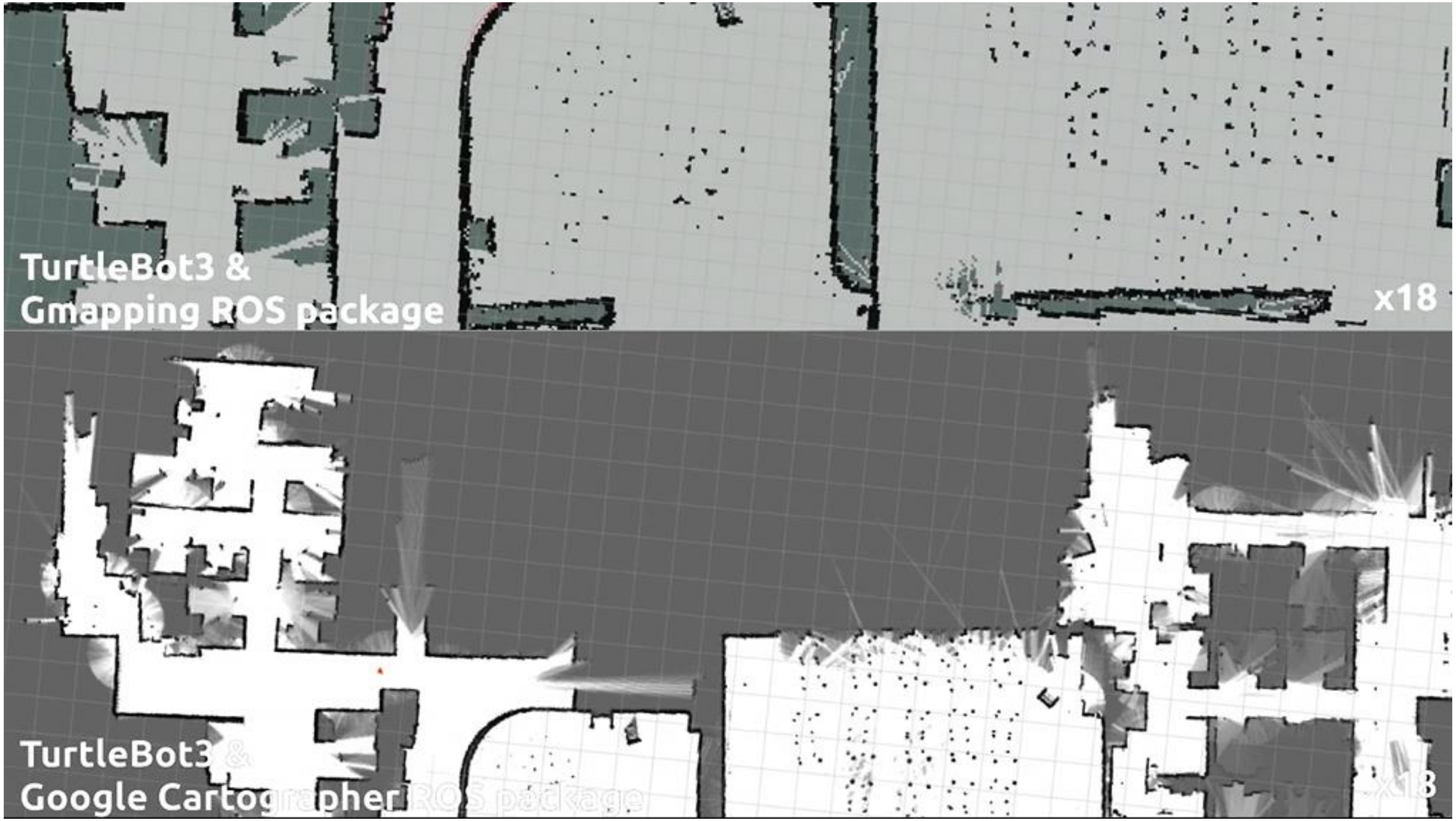


2D **Occupancy Grid Map (OGM)**

- White = **Free area** where robot can move
- Black = **Occupied area** where robot can not move
- Gray = **Unknown area**

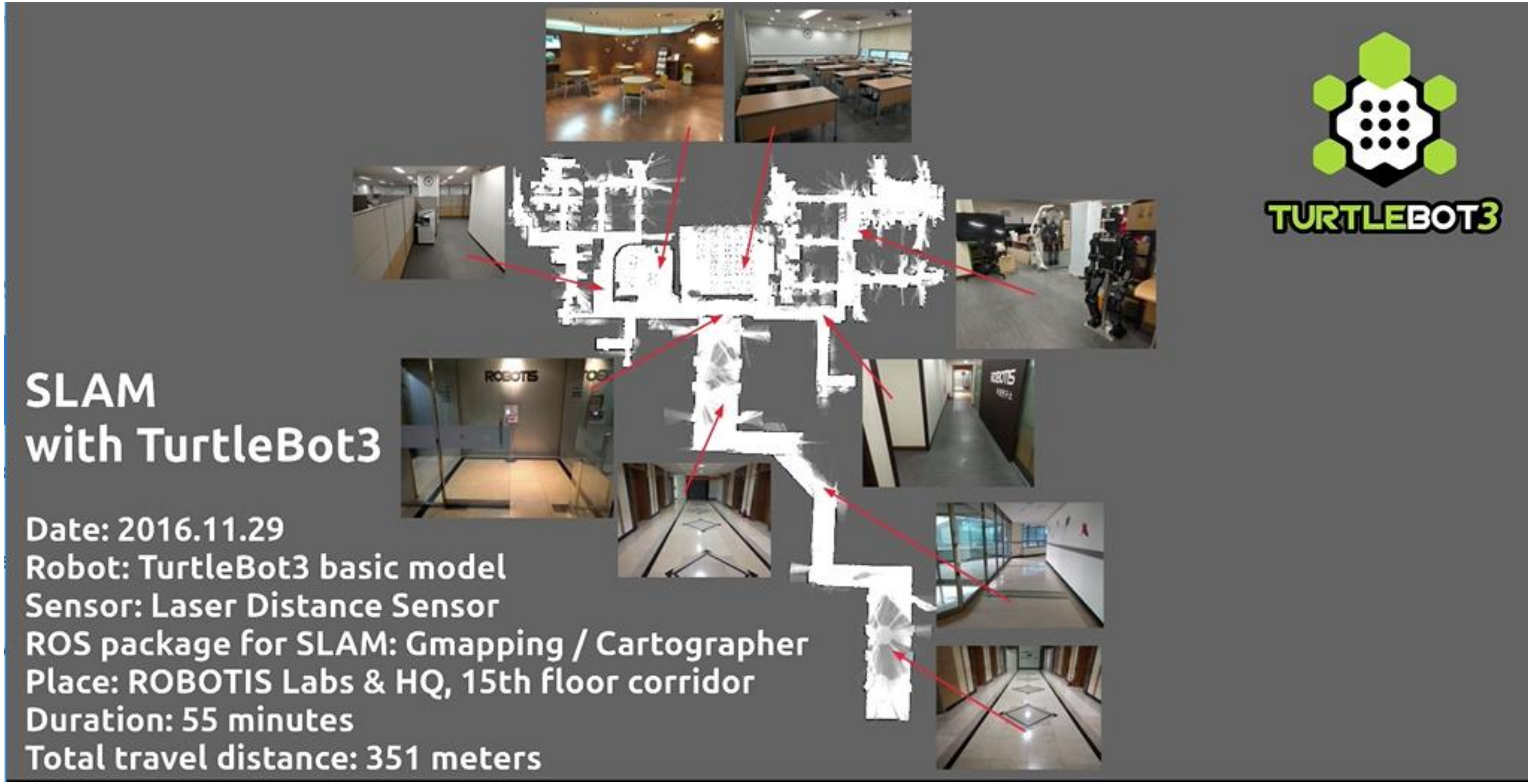


Mapping: Gmapping & Cartographer + TurtleBot3



https://youtu.be/lkW4-dG2BCY

Mapping: Gmapping & Cartographer + TurtleBot3



https://youtu.be/lkW4-dG2BCY

Mapping

How about it, easy?

Mapping

If you want to develop a service or mobile robot with SLAM & Navigation as basic function,

Just use SLAM and Navigation as they are and spend more time in your desired area!

I look forward to seeing your unique robot that is not in the world

How about it, easy?

Mapping

~~***How about it, easy?***~~

If you want to develop a service or mobile robot with SLAM & Navigation as basic function,
Just use SLAM and Navigation as they are and spend more time in your desired area!
I look forward to seeing your unique robot that is not in the world

If you want to study more about SLAM & Navigation,
All software is opened.
See it , understand it, and add functions!
There is no better textbook than practice

Process of SLAM Related Nodes

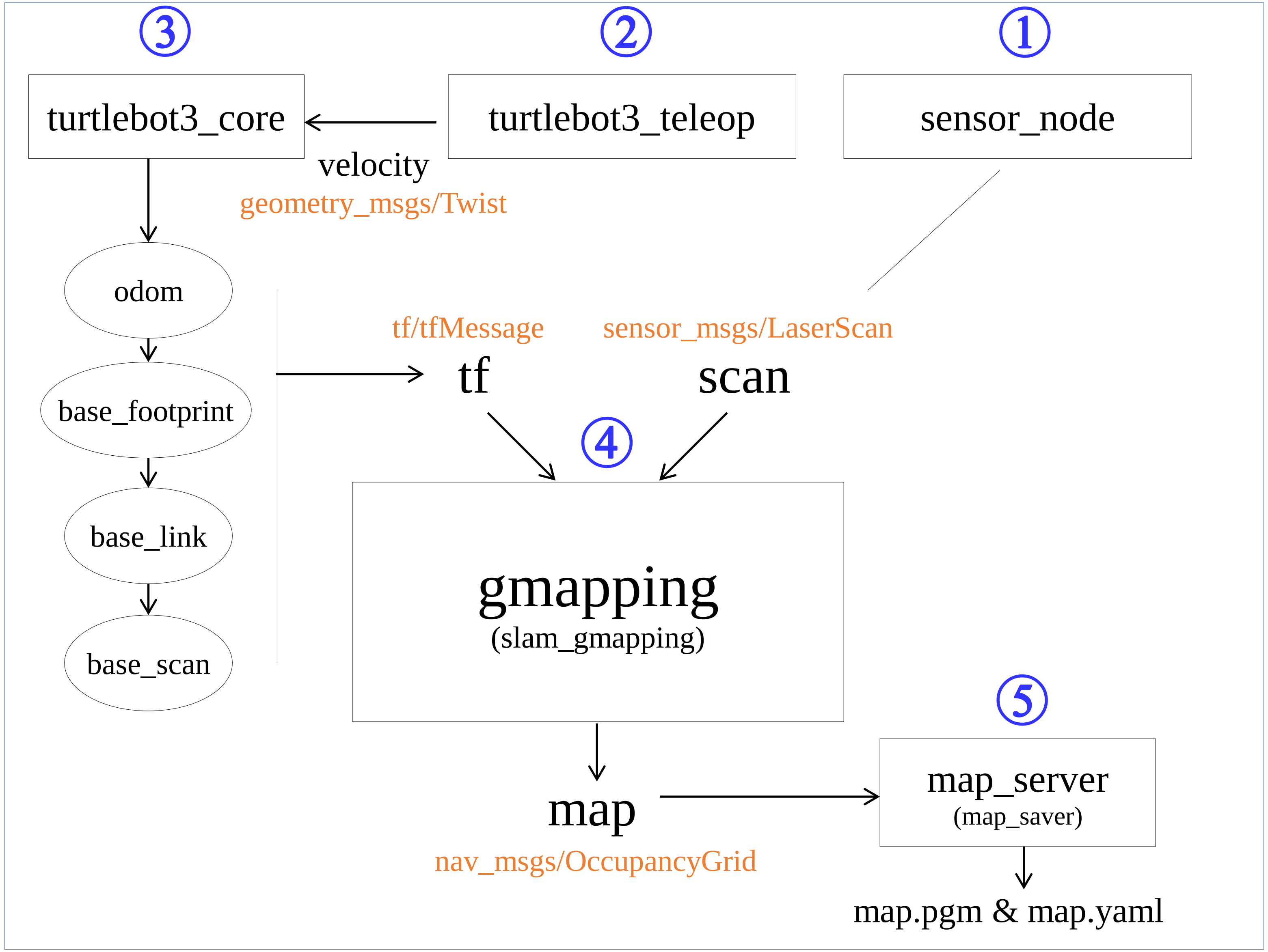
① **sensor_node**

② **turtlebot3_teleop**

③ **turtlebot3_core**

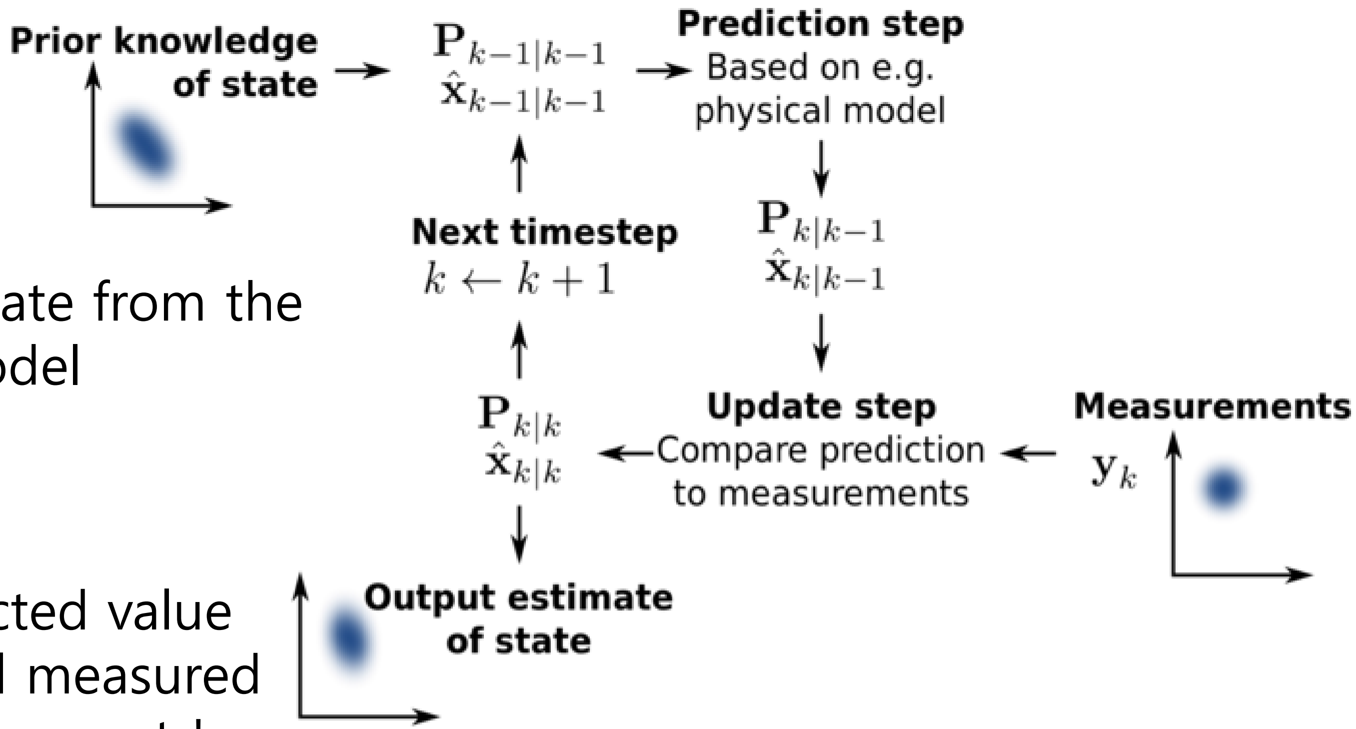
④ **slam_gmapping**

⑤ **map_server**



Localization | Kalman filter, Particle filter, Graph, Bundle adjustment

- **Kalman filter**
  - A recursive filter that tracks the state of an object in a linear system containing noise
  - Based on Bayesian probability

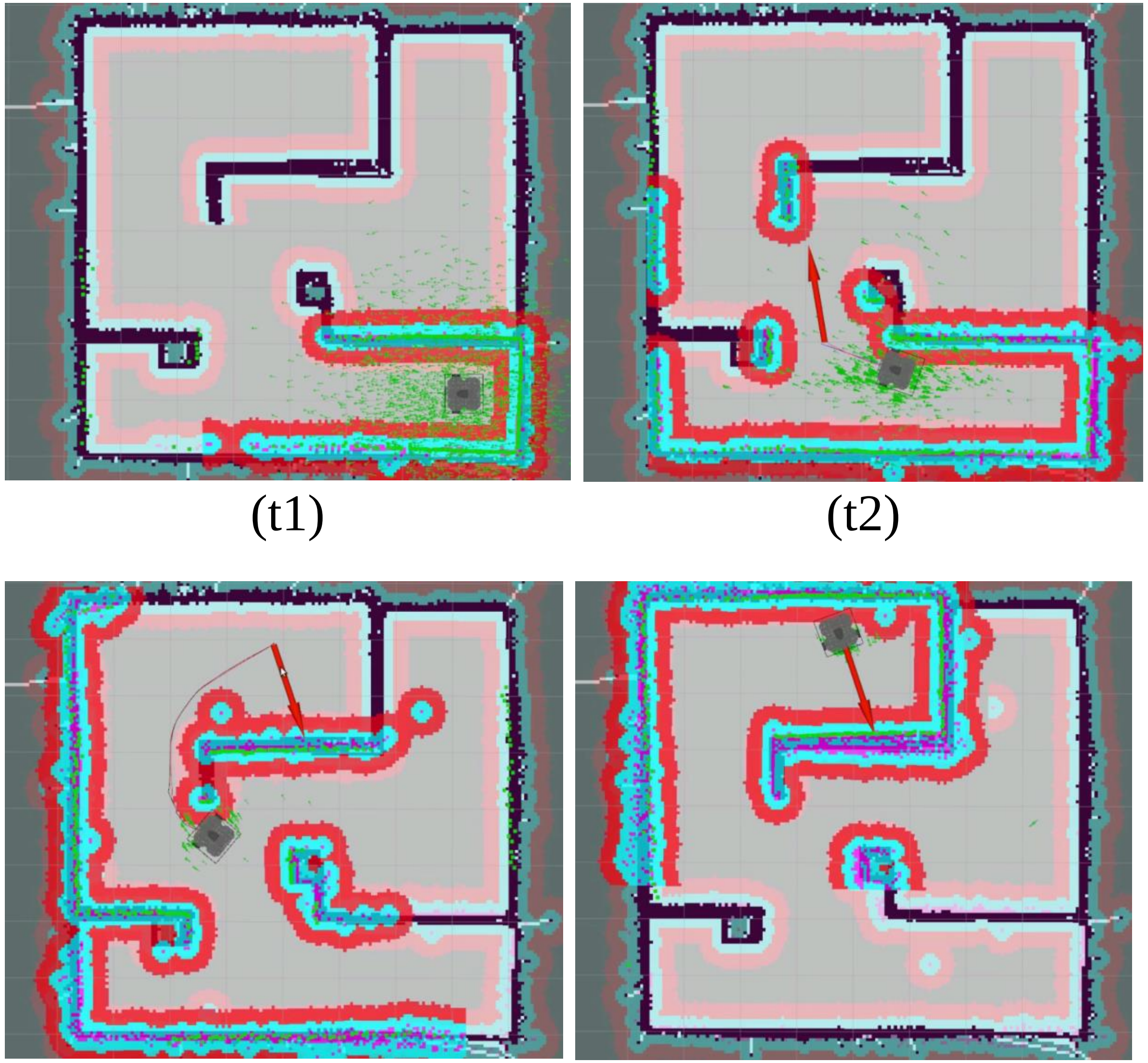


  - Prediction
    - Estimate the state of the current state from the previous state by assuming the model
  - Update
    - Using the error between the predicted value of the previous step and the actual measured value obtained by the external instrument !

Localization | Kalman filter, Particle filter, Graph, Bundle adjustment

- Particle Filter
- A particle filter is a technique that predicts through simulation based on a trial-and-error method. Estimated values are represented as particles with probabilistic representation.

1) **Initialization**
2) **Prediction**
3) **Update**
4) **Pose Estimation**
5) **Resampling**



(t1) (t2) (t3) (t4)

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

Navigation

Navigation + TurtleBot3

- http://turtlebot3.robotis.com/en/latest/navigation.html
- Run Master (Remote PC)

```
$ roscore
```

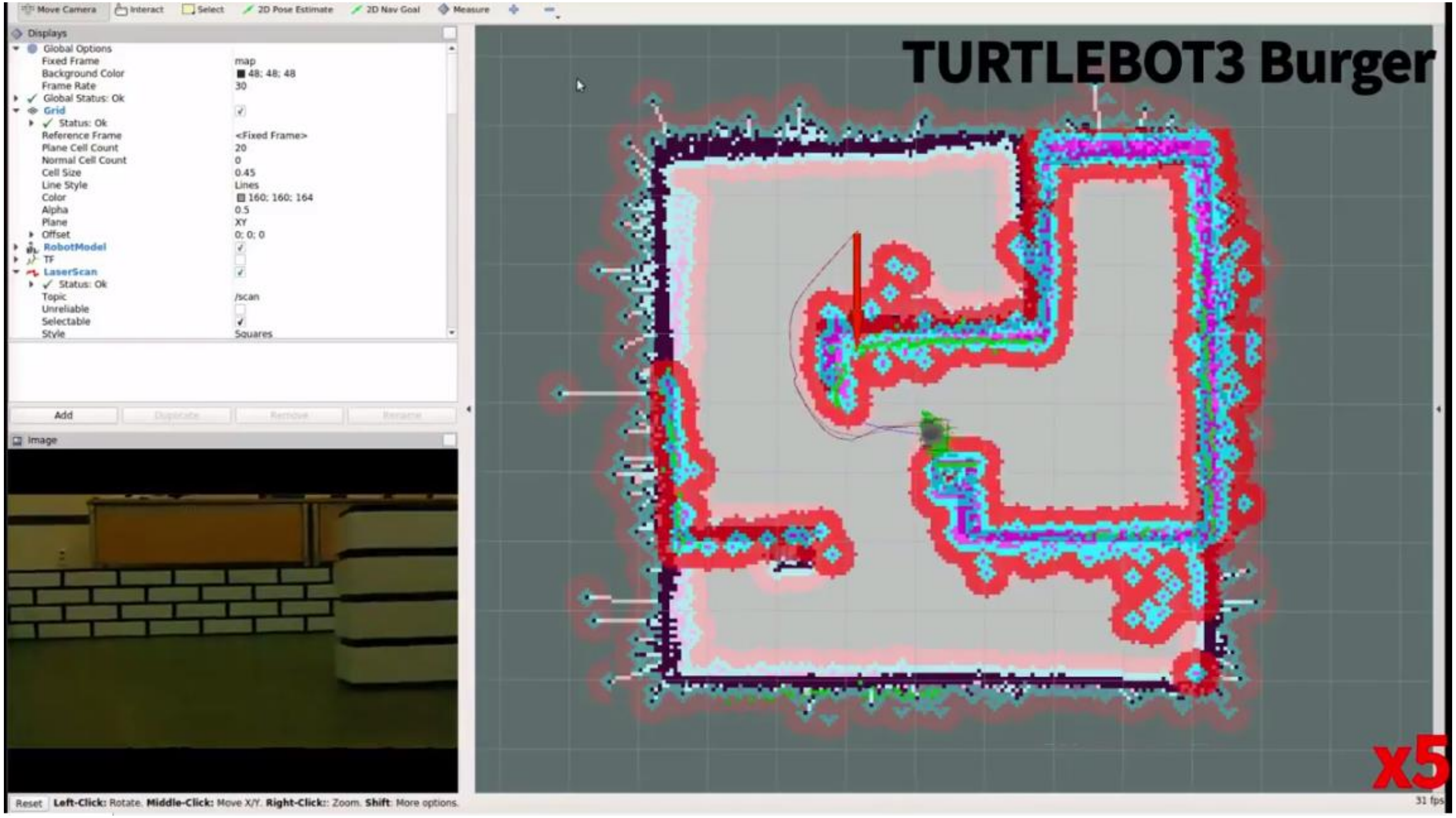
- TurtleBot3 and Sensor Drive (SBC)

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

- RViz, TurtleBot3 Remote Control, Navigation (Remote PC)

```
$ export TURTLEBOT3_MODEL=burger  (or waffle or waffle_pi)
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
$ rosrun rviz rviz -d `rospack find turtlebot3_navigation`/rviz/turtlebot3_nav.rviz
```

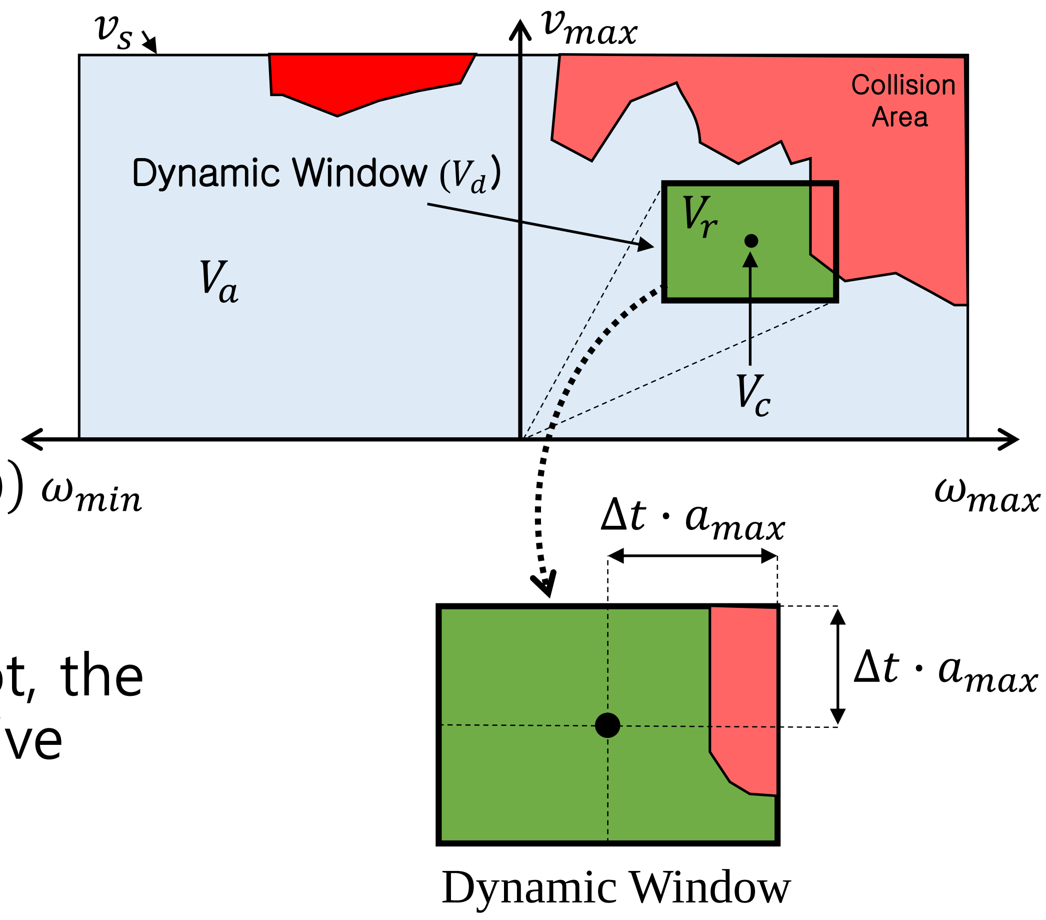
Navigation



https://youtu.be/VYlMywwYALU

Navigation

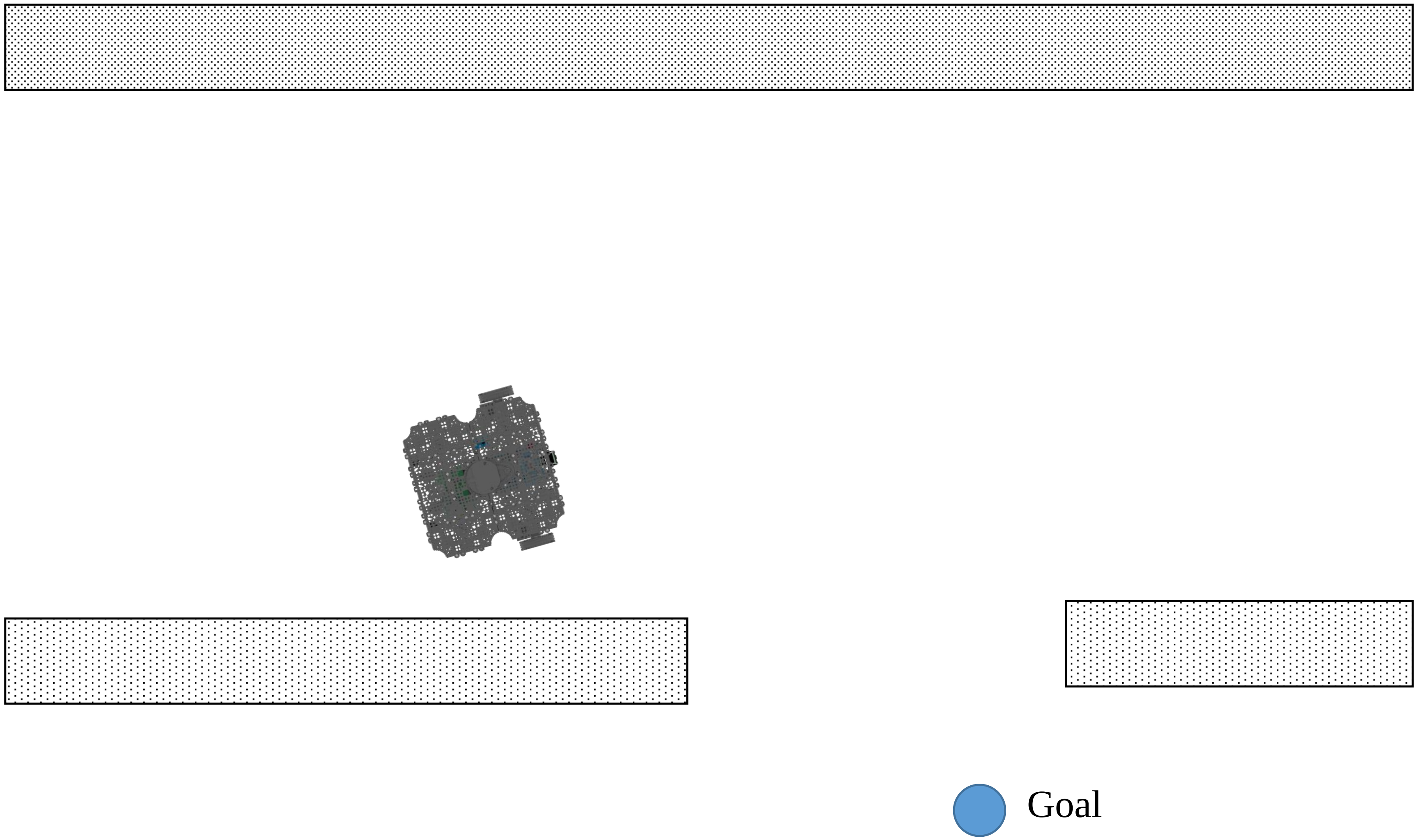
- **Dynamic Window Approach** (Mainly used in local plan)
- How to choose the speed at which you can quickly reach the target point while avoiding the obstacles that can collide with the robot in the 'velocity search space' of the robot
    - v (linear velocity), ω (angular velocity)
    - V_s: Available speed range
    - V_a: Permissible speed range
    - V_r: Speed range in dynamic window
    - $G_{(v,\omega)} = \sigma\big(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega)\big)$



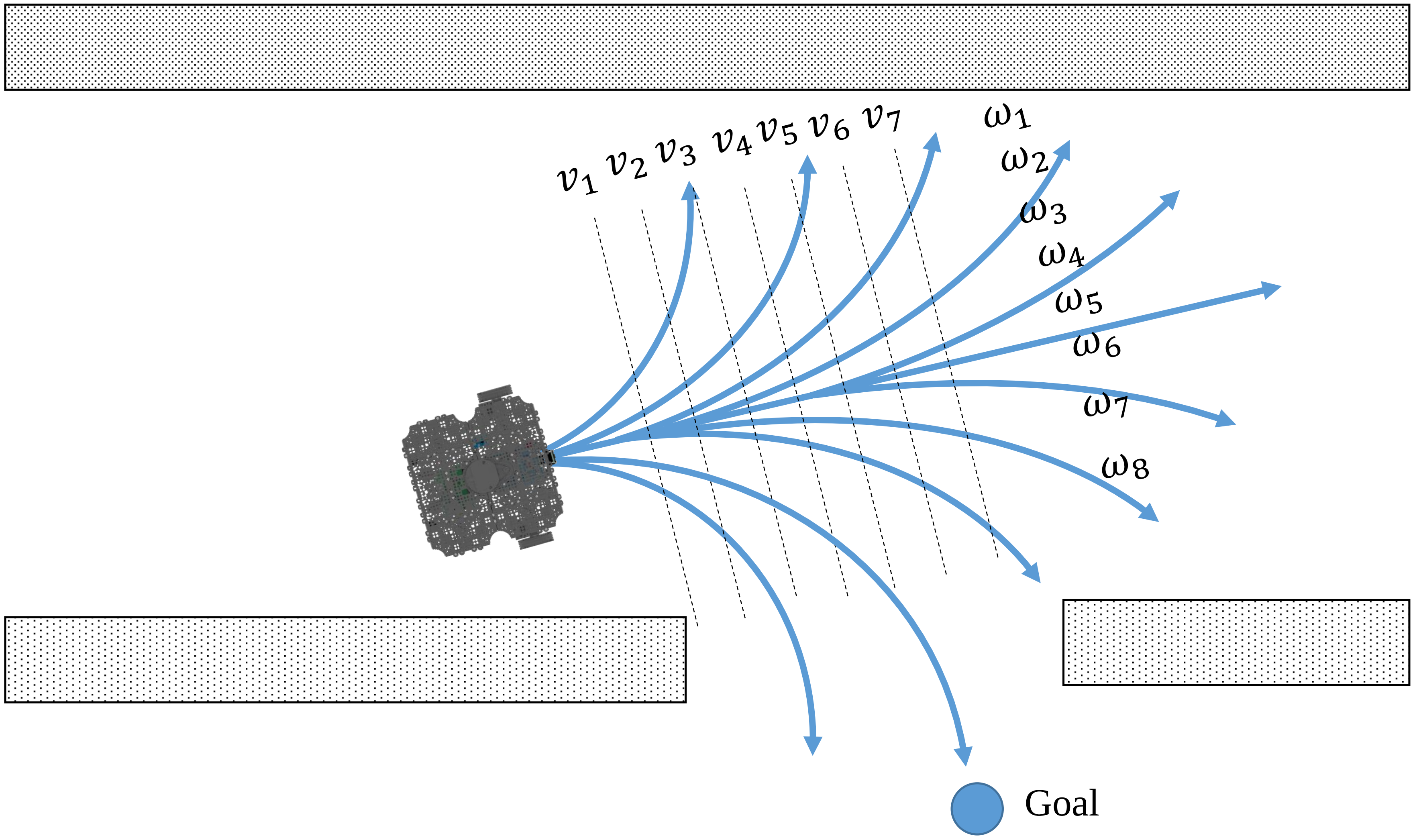
- Given the direction, speed, and impact of the robot, the objective function G finds v, ω at which the objective function is maximized

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

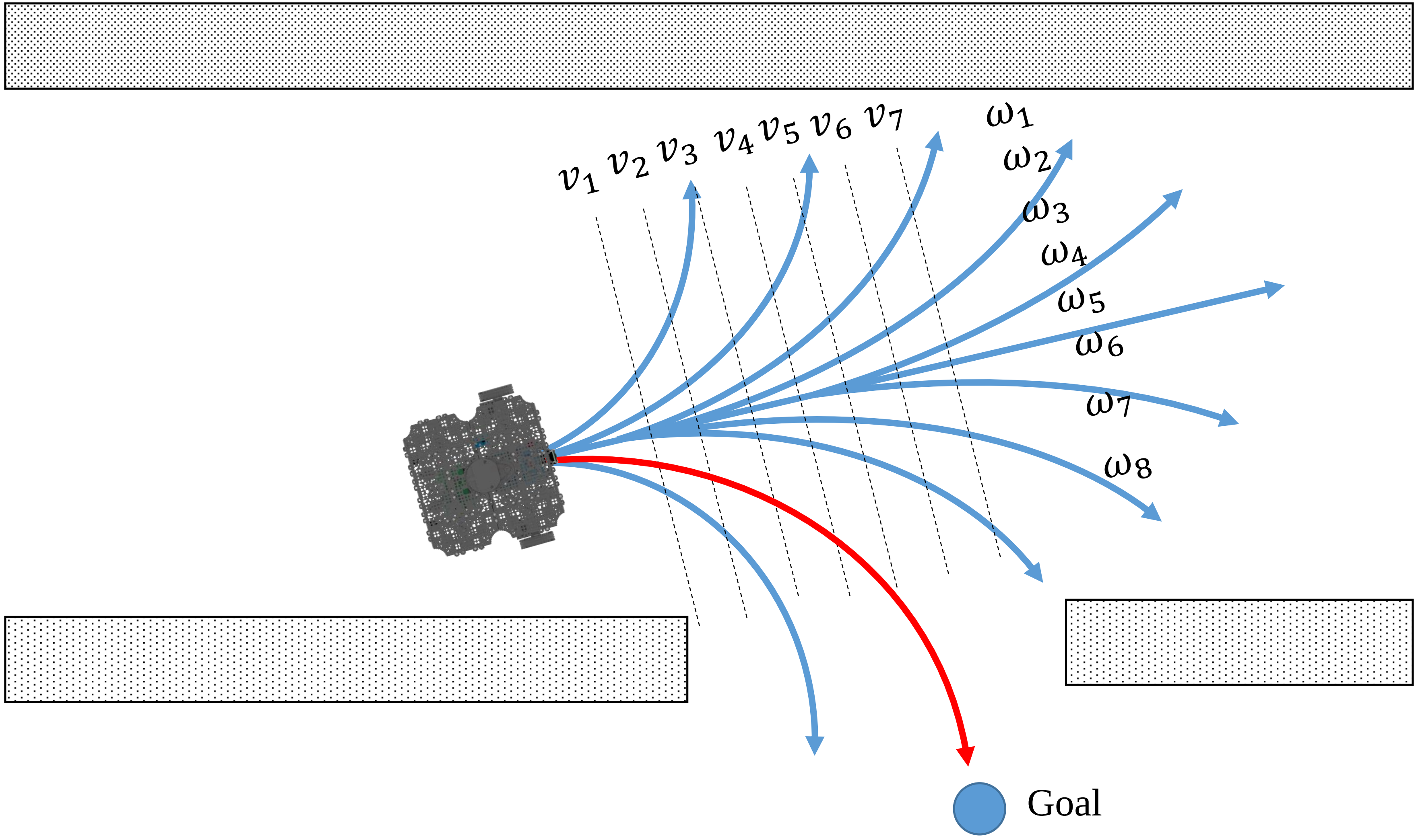
Dynamic Window Approach (DWA)



Dynamic Window Approach (DWA)



Dynamic Window Approach (DWA)



Mapping

How about it, easy?

If you want to develop a service or mobile robot with SLAM & Navigation as basic function,
Just use SLAM and Navigation as they are and spend more time in your desired area!
I look forward to seeing your unique robot that is not in the world

If you want to study more about SLAM & Navigation,
All software is opened.
See it , understand it, and add functions!
There is no better textbook than practice

Practice Time!

"SLAM / Navigation"

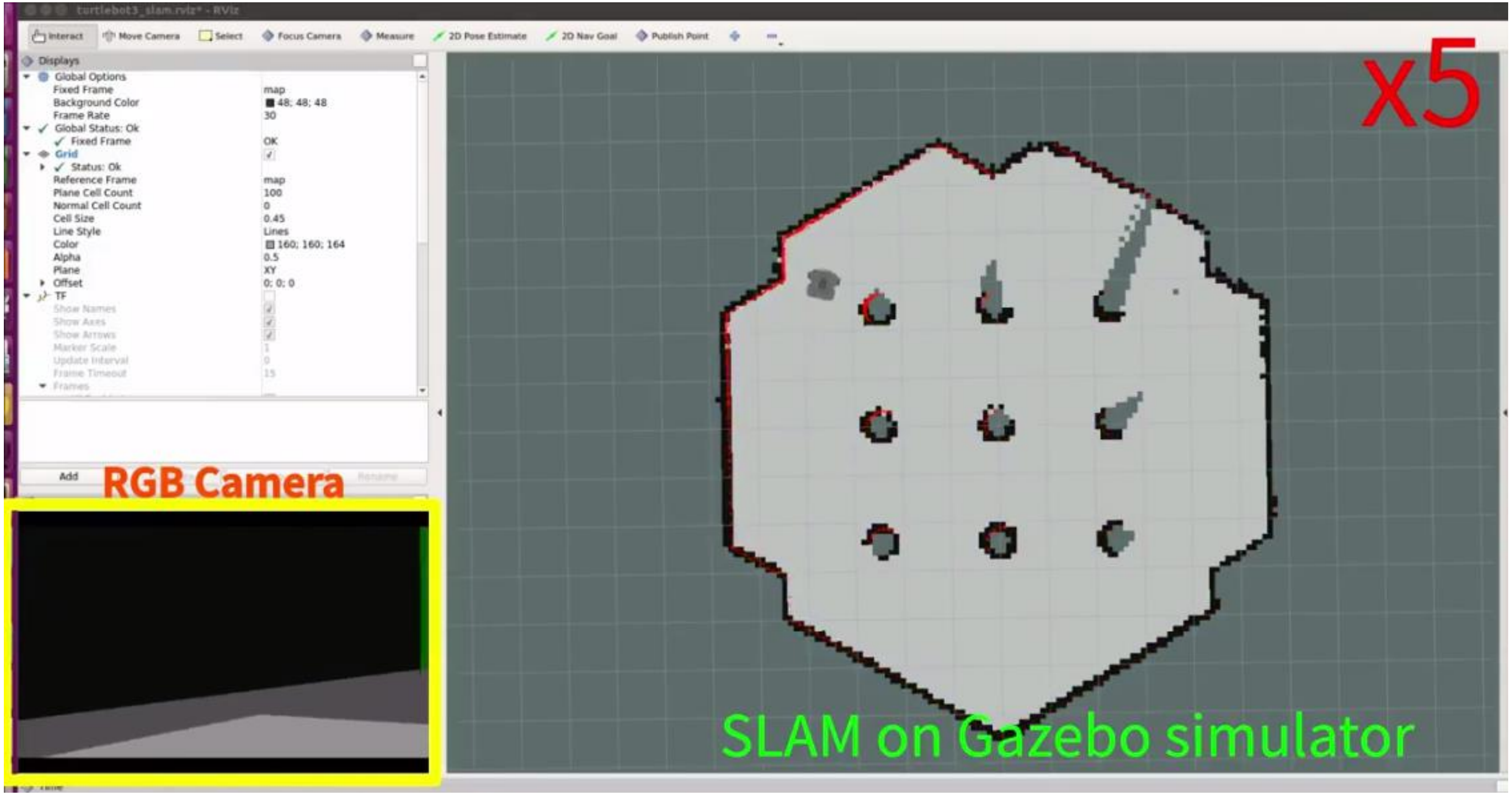
Preparation of TurtleBot3 Simulation Development Environment

- Official TurtleBot3 wiki reference
  - http://turtlebot3.robotis.com
- Basic package installation (for the use of 3D simulator Gazebo)

```
$ sudo apt-get 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

Run Virtual Robot with Gazebo

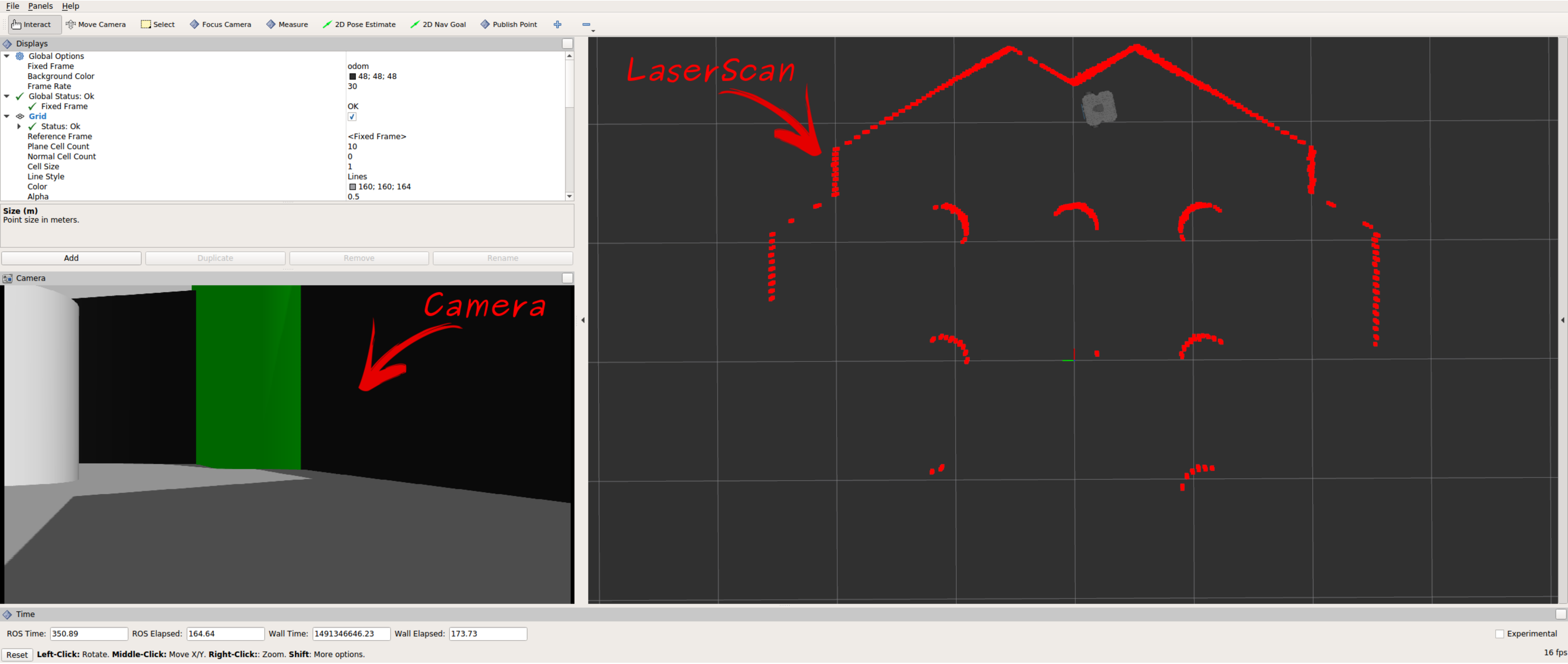
- Control virtual robot on 3D simulator Gazebo
  - Robots can be controlled with 'Turtlebot3_teleop_key' node
  - Possible to check sensor value mounted on robot on Gazebo via Rviz
    - 2D Laser Range Sensor, Camera, Depth Camera, IMU, etc.

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

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

```
$ export TURTLEBOT3_MODEL=waffle_pi
$ roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch
```

Run Virtual Robot with Gazebo



Virtual SLAM with Gazebo

- **Run Gazebo**

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

- **Run SLAM**

```
$ export TURTLEBOT3_MODEL=waffle_pi
$ roslaunch turtlebot3_slam turtlebot3_slam.launch
```

- **Run RViz**

```
$ export TURTLEBOT3_MODEL=waffle_pi
$ rosrun rviz rviz -d `rospack find turtlebot3_slam`/rviz/turtlebot3_slam.rviz
```

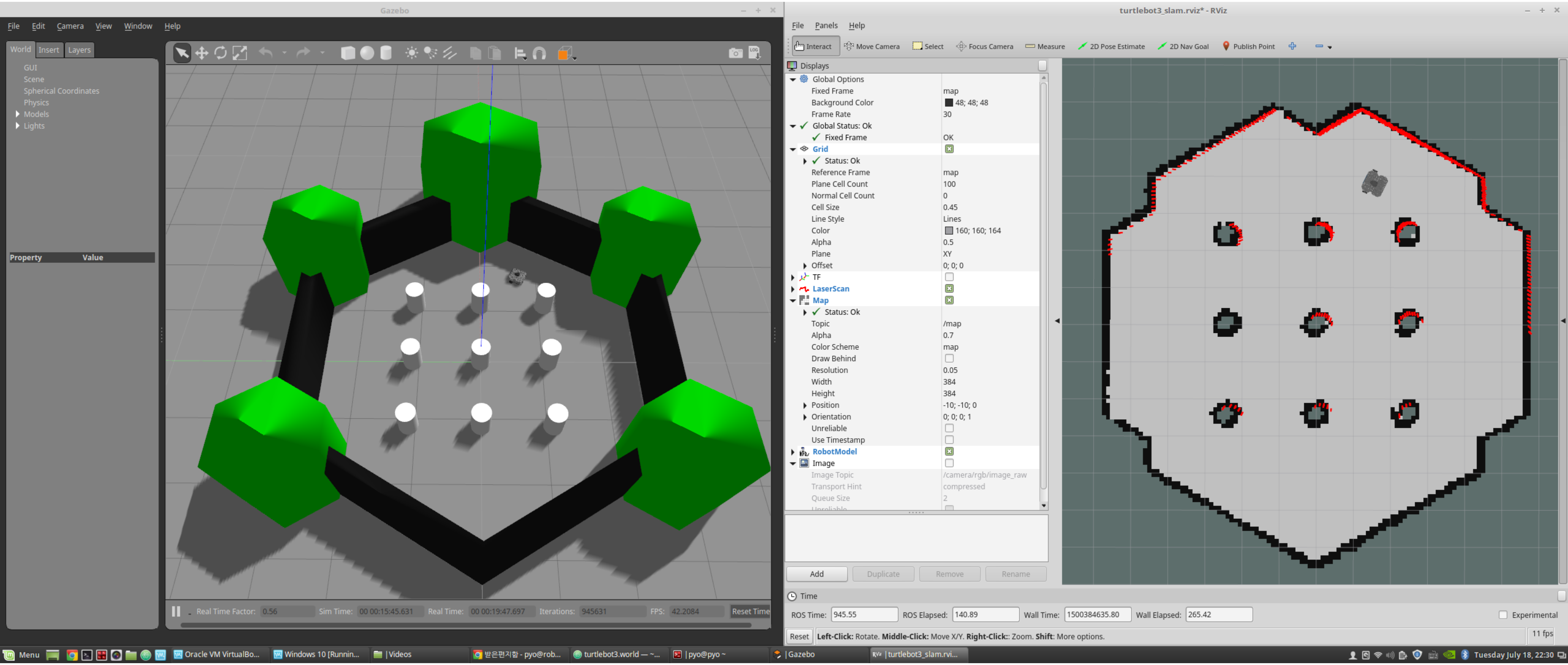
- **TurtleBot3 Remote Control**

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

- **Run Map Server**

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

Virtual SLAM with Gazebo



Virtual Navigation with Gazebo

- **Run Gazebo**

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

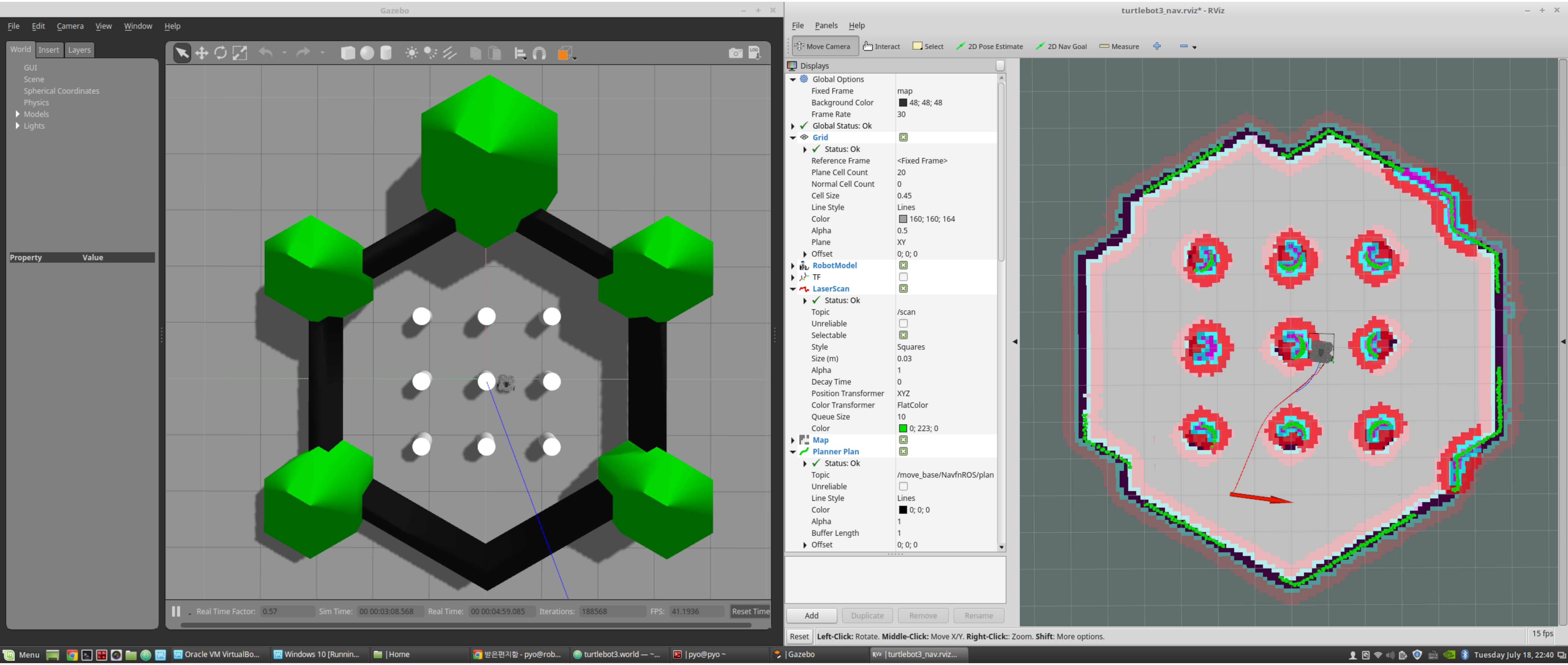
- **Run Navigation**

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

- **Run RViz & Setting Destination**

```
$ export TURTLEBOT3_MODEL=waffle_pi
$ rosrun rviz rviz -d `rospack find turtlebot3_navigation`/rviz/turtlebot3_nav.rviz
```

Virtual Navigation with Gazebo



Question Time!

Advertisement #1

Download link

Language:

English, Chinese, Japanese, Korean

"ROS Robot Programming"

A Handbook is written by TurtleBot3 Developers

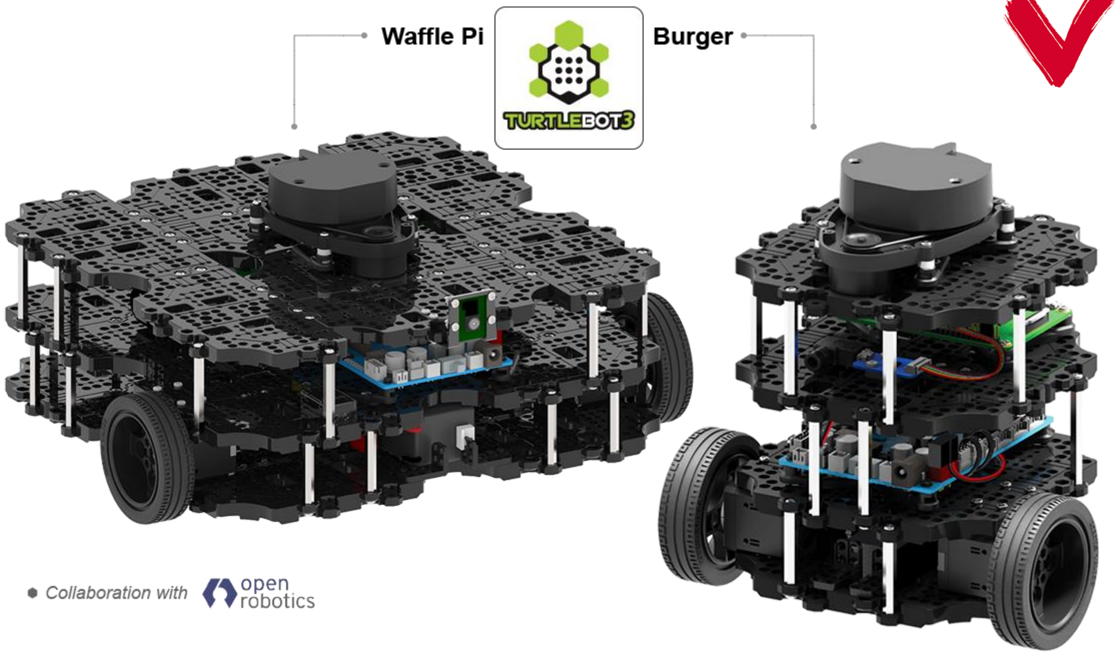
Advertisement #2

TURTLEBOT3

AI Research Starts Here
ROS Official Platform

TurtleBot3 is a new generation mobile robot that’s modular, compact and customizable. Let's explore ROS and create exciting applications for education, research and product development.

Direct Link



Advertisement #3

Robot Source

www.robotsource.org

The ‘RobotSource’ community is the space for people making robots.

We hope to be a community where we can share knowledge about robots, share robot development information and experiences, help each other and collaborate together. Through this community, we want to realize open robotics without disti nguishing between students, universities, research institutes and companies.

Join us in the Robot community ~

END.