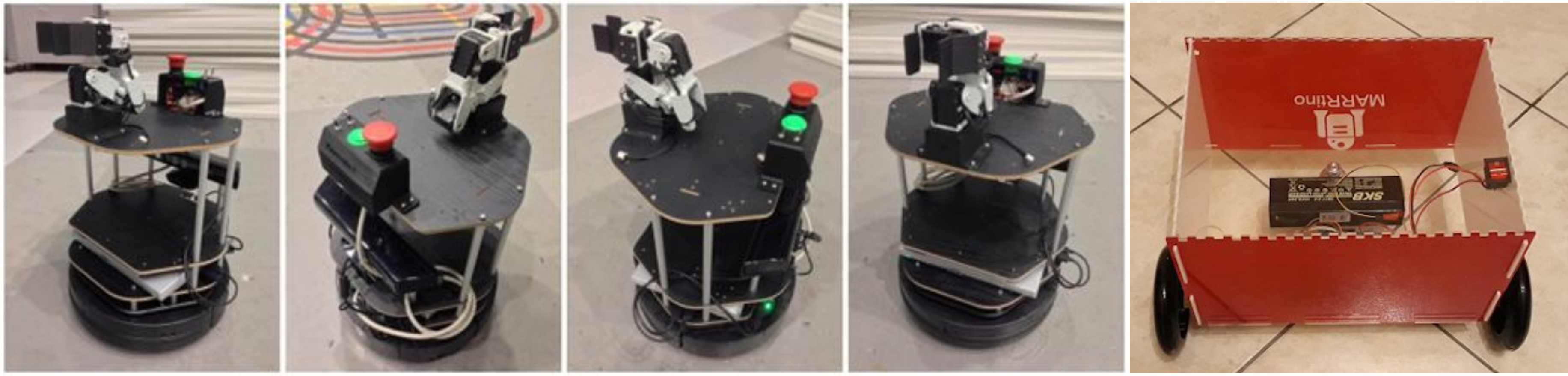


RoboCup@Home Education
ONLINE CHALLENGE 2020
Online Classroom Open Platform

05 Robot Navigation (1/2) : SLAM Map Building (Simulation)

Jeffrey Tan, Jupiter Robot | 2020.05.14

RoboCup@Home EDUCATION

MathWorks®

JUPITER ROBOT

Online Challenge 2020: Online Classroom OP

05 Robot Navigation (1/2) : SLAM Map Building (Simulation)

Speakers: Jeffrey Tan, Jupiter Robot

Time: **May 14, 2020 (Thu) 10:00 - 11:00 am (GMT+8)**

05 Robot Navigation (2/2) : Autonomous Navigation (Simulation)

Speakers: Jeffrey Tan, Jupiter Robot

Time: **May 14, 2020 (Thu) 11:00 - 12:00 noon (GMT+8)**

Zoom: https://cernet.zoom.com.cn/j/63946172707 | PW: robocup

Facebook Live: https://www.facebook.com/robocupathomeedu/live/

Web:
https://www.robocupathomeedu.org/challenges/robocuphome-education-online-challenge-2020

Online Classroom:
https://www.robocupathomeedu.org/learn/online-classroom/online-challenge-2020

** Privacy reminder: Video will be recorded and published online.

Previous Assignment / Prerequisites

1. Combine speech interaction and visual perception to develop an interactive robot vision application
   a. Design and develop robot vision application using human/object detection/tracking with speech interface.
2. Upload to GitHub
   a. Create own repository and upload the source code, system design, and visual+speech interaction video (with terminal results) to GitHub.

Robot Navigation Development System (Simulation)

Hardware

- Laptop computer

Software

- Ubuntu
- ROS
- Gazebo (simulator)
- Related component software

Robot Navigation

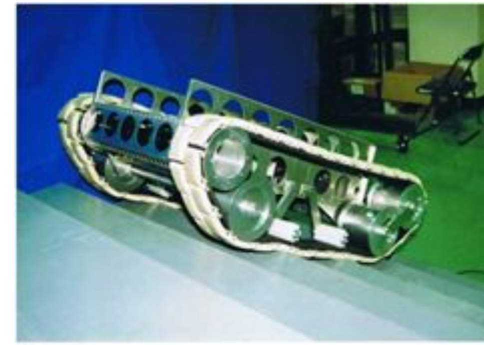
SLAM Map Building

- Mobile Robot
- Robot Odometry and Localization

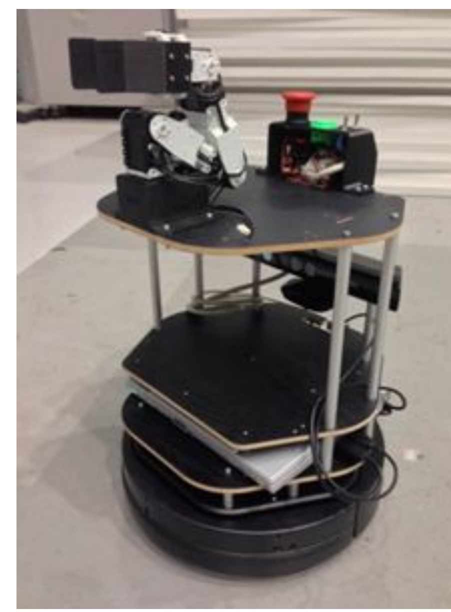
Autonomous Navigation

- Robot Motion Planning
- Autonomous Navigation

Mobile Robot
Mobility Design & Principle

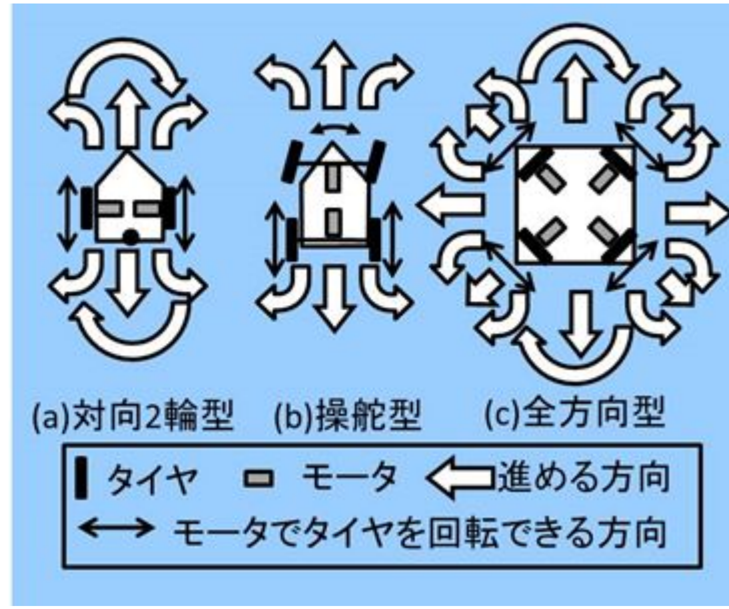


- 2 legged robot
- Multi legged robot
- Wheel
- Track
- Omnidirectional
- Crawling
- Wall climbing
- Leg-wheel hybrid



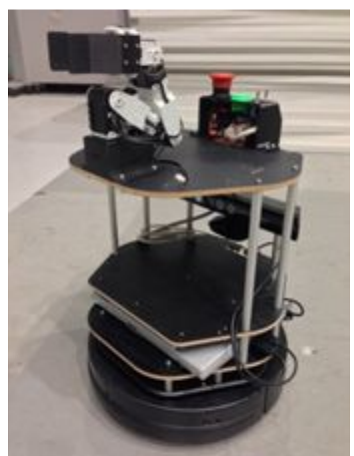
Mobile Robot
Wheeled Mobile Robot Mechanism

1. Parallel 2-wheel
2. Ackermann steering
3. Omnidirectional



Holonomic (c)
(*x*,*y*,θ) of all degree-of-freedom can be controlled independently

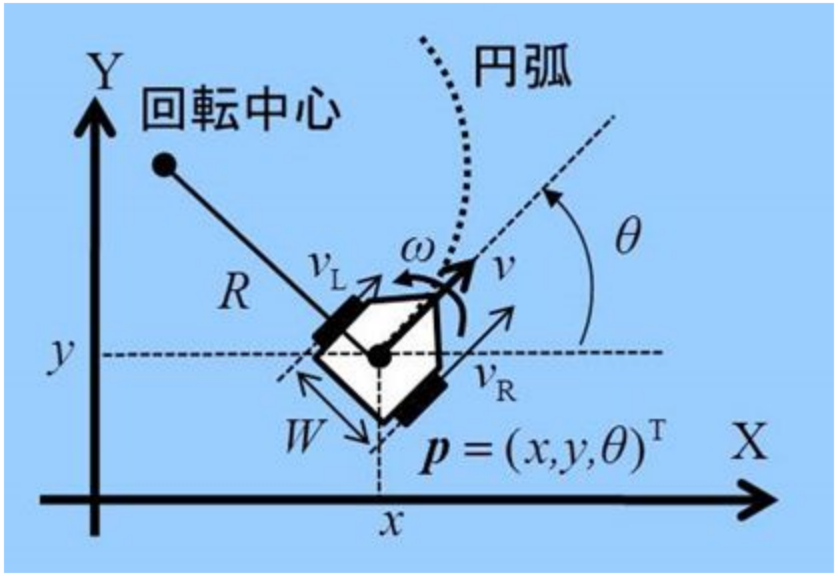
Non-holonomic (a,b)
(*x*,*y*,θ) can't be controlled independently



Mobile Robot
Control of Wheeled Mobile Robot

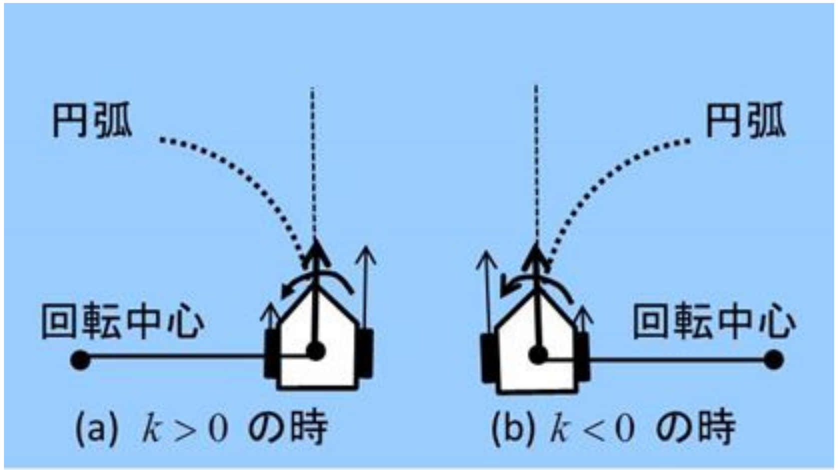
Modeling of parallel 2-wheel mobile robot

- v_L left wheel speed
- v_R right wheel speed
- R turning radius
- W wheel track



Curvature and center of rotation

- Curvature k=1/R



Mobile Robot
Control of Wheeled Mobile Robot

Left and right wheel speed, robot travel speed and angular velocity

$$v_L = \left(R - \frac{W}{2}\right)\omega \qquad (1)$$

$$v_R = \left(R + \frac{W}{2}\right)\omega \qquad (2)$$

$$v = R\omega \qquad (3)$$

$$v = \frac{v_R + v_L}{2} \qquad (4)$$

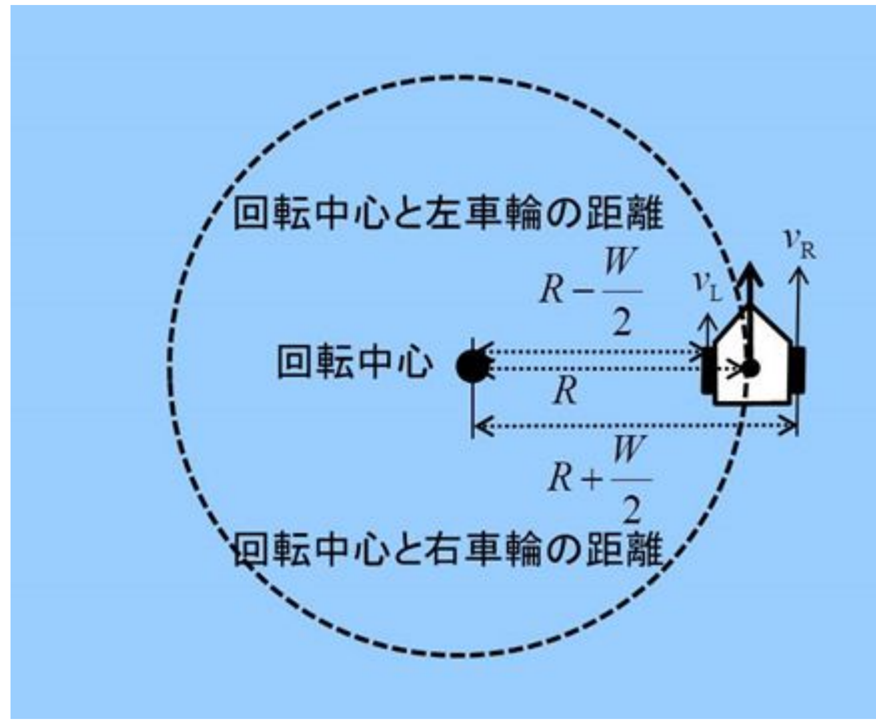
$$\omega = \frac{v_R - v_L}{W} \qquad (5)$$

$$v_R = v + \frac{W}{2}\omega \qquad (6)$$

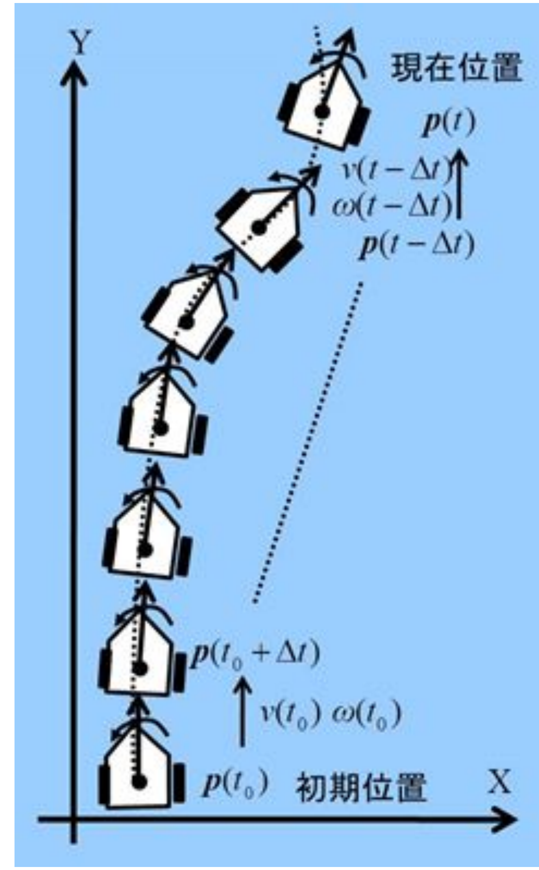
$$v_L = v - \frac{W}{2}\omega \qquad (7)$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ 1/W & -1/W \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} \qquad (8)$$

$$\begin{bmatrix} v_R \\ v_L \end{bmatrix} = \begin{bmatrix} 1 & W/2 \\ 1 & -W/2 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \qquad (9)$$



Mobile Robot
Position and Orientation Estimations



Odometry

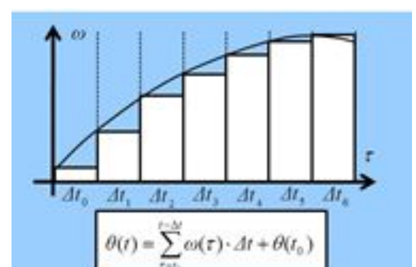
$$\begin{aligned} x(t) &= \int_{t_0}^{t} v\cdot cos\theta(\tau)d\tau + x(t_0) \\ y(t) &= \int_{t_0}^{t} v\cdot sin\theta(\tau)d\tau + y(t_0) \\ \theta(t) &= \int_{t_0}^{t} \omega(\tau)d\tau + \theta(t_0) \end{aligned} \qquad (10)$$

Strip approximation

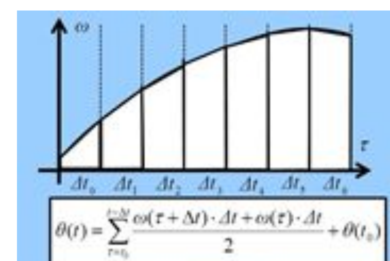
$$\begin{aligned} x(t) &= \sum_{\tau=t_0}^{t-\Delta t} v(\tau)\Delta t\cdot cos\theta(\tau) + x(t_0) \\ y(t) &= \sum_{\tau=t_0}^{t-\Delta t} v(\tau)\Delta t\cdot sin\theta(\tau) + y(t_0) \\ \theta(t) &= \sum_{\tau=t_0}^{t-\Delta t} \omega(\tau)\Delta t + \theta(t_0) \end{aligned} \qquad (11)$$

Trapezoid approximation

$$\begin{aligned} x(t) &= \sum_{\tau=t_0}^{t-\Delta t} \frac{v(\tau+\Delta t)+v(\tau)}{2}\Delta t\cdot cos\theta(\tau) + x(t_0) \\ y(t) &= \sum_{\tau=t_0}^{t-\Delta t} \frac{v(\tau+\Delta t)+v(\tau)}{2}\Delta t\cdot sin\theta(\tau) + y(t_0) \\ \theta(t) &= \sum_{\tau=t_0}^{t-\Delta t} \frac{\omega(\tau+\Delta t)+\omega(\tau)}{2}\Delta t + \theta(t_0) \end{aligned} \qquad (12)$$



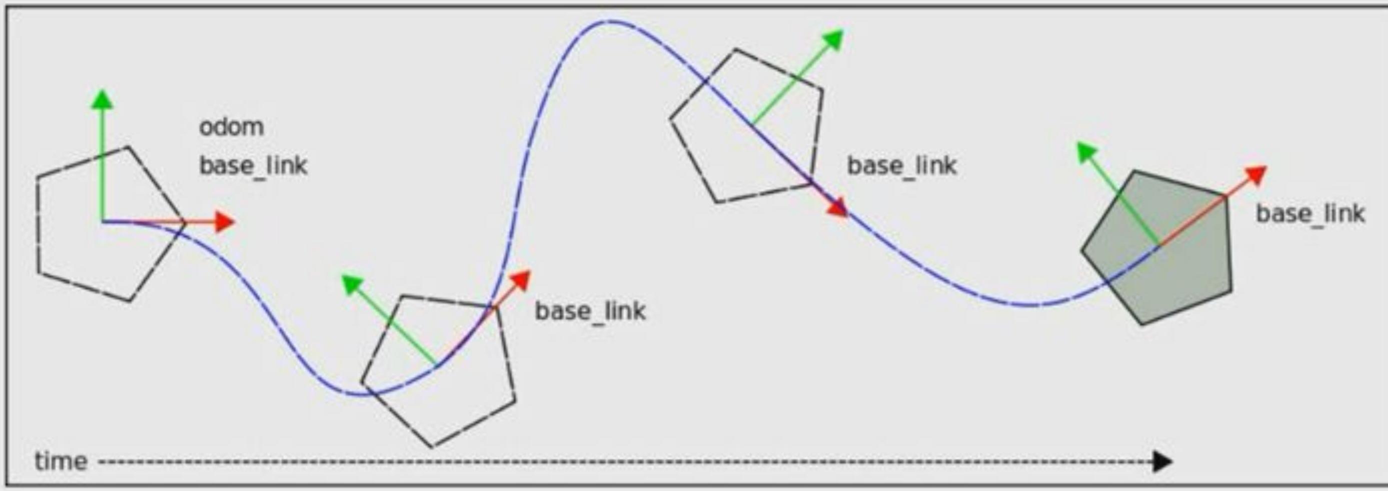
(a) 短冊近似



(b) 台形近似

Robot Odometry and Localization: TurtleBot Odometry

[https://www.youtube.com/watch?v=3S8MXsnNe3U]

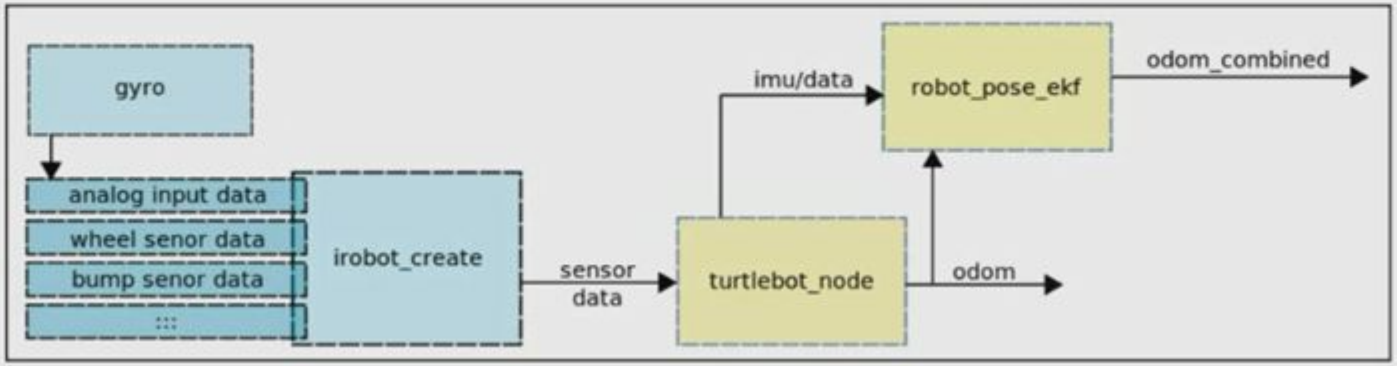


Odometry is the use of data from moving sensors to estimate change in position over time. Odometry is used by some robots, whether legged or wheeled, to estimate (not determine) their position relative to a starting location.

http://en.wikipedia.org/wiki/Odometry

Robot Odometry and Localization: TurtleBot Odometry

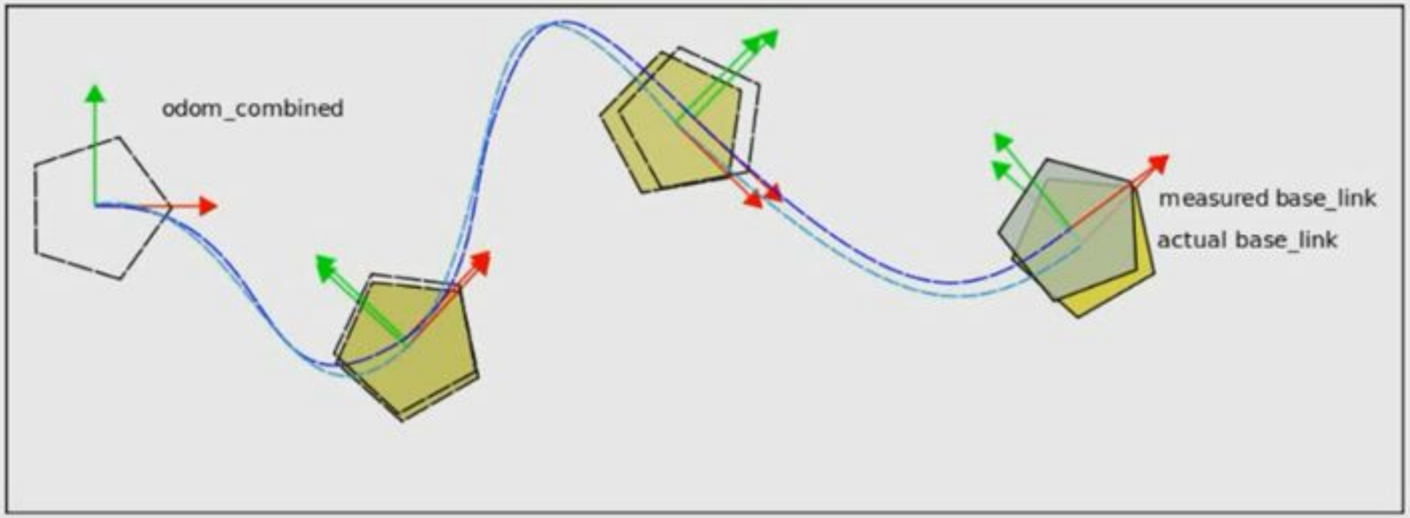
[https://www.youtube.com/watch?v=3S8MXsnNe3U]



To improve the TurtleBot odometry we have added a gyro to the irobot create. The robot_pose_ekf node use the gyro and odom data to compute and publish the more accurate odom_combined.

Robot Odometry and Localization: TurtleBot Odometry

[https://www.youtube.com/watch?v=3S8MXsnNe3U]

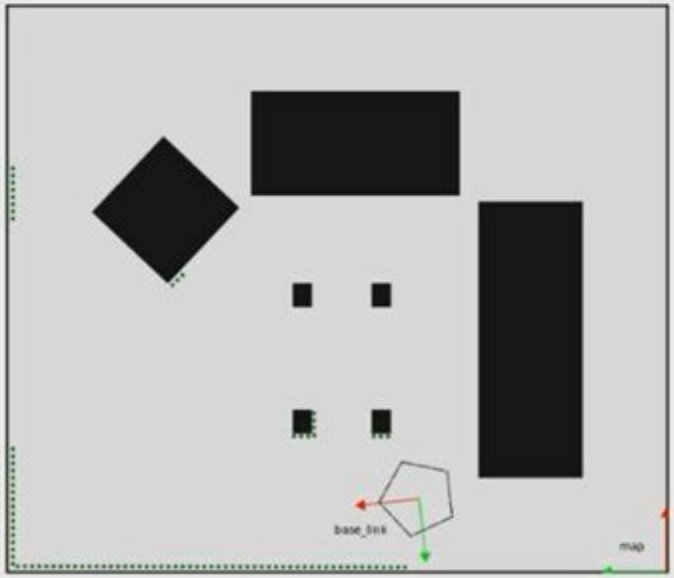


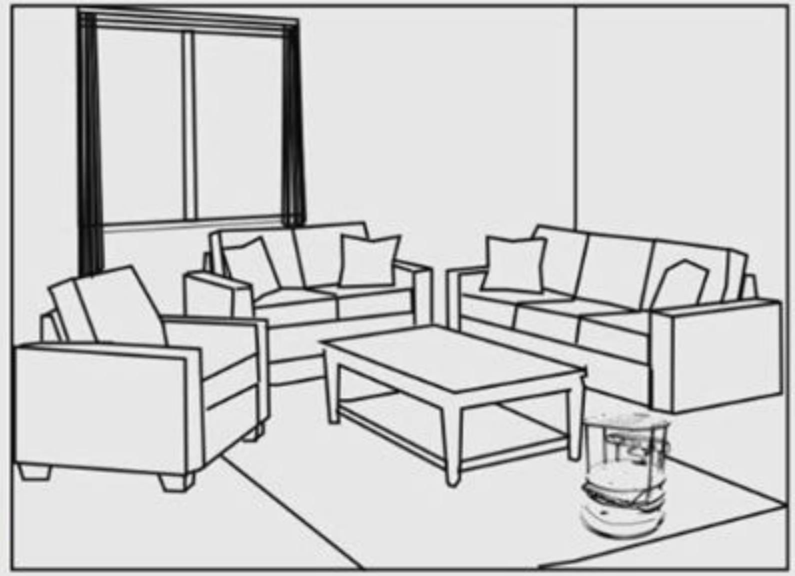
robot_pose_ekf (ekf: extended kalman filter) uses measurements observed over time, containing noise (random variations) and other inaccuracies, and produce values that tend to be closer to the true values of the measurements than their associated calculated values.

http://en.wikipedia.org/wiki/Kalman_filter

Robot Odometry and Localization: TurtleBot Localization

[https://www.youtube.com/watch?v=Mv1mbsMfbmI]

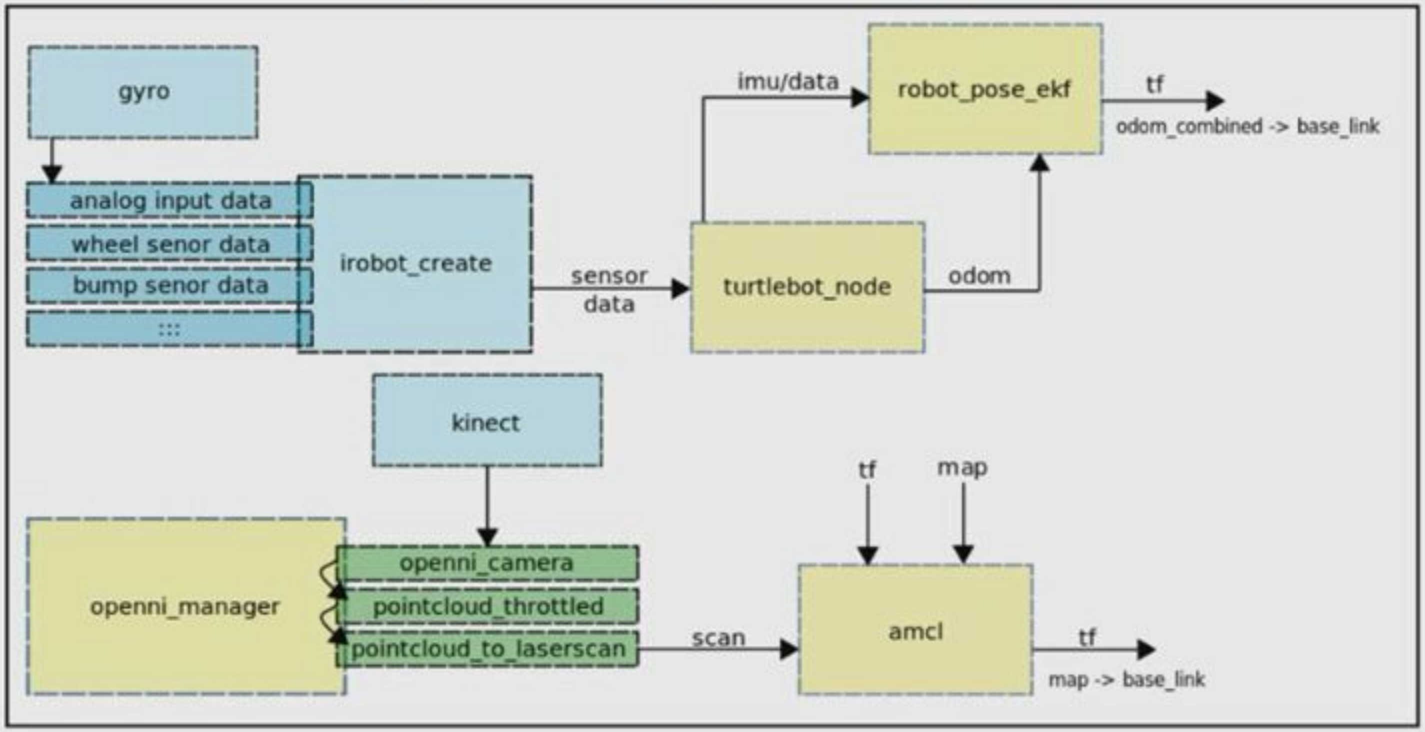




In your home a robot localizes itself using odometry and laser scan data. The right image shows a map of the image on the left with robot localized with laser data overlayed on the map image. The grey areas of the map show the unobstructed areas of the map while the black show the obstructed areas.

Robot Odometry and Localization: TurtleBot Localization

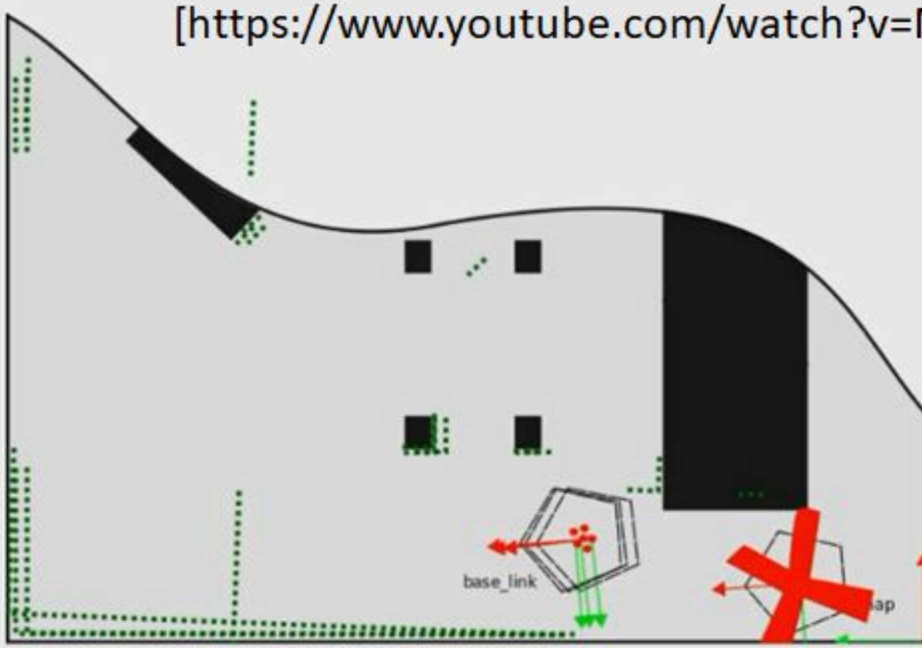
[https://www.youtube.com/watch?v=Mv1mbsMfbmI]



The openni_manager uses the kinect pointcloud data to create laser scan data for use with amcl. The amcl node uses the scan data and odom_combined to compute the transform from the map to base_link.

Robot Odometry and Localization: TurtleBot Localization

[https://www.youtube.com/watch?v=Mv1mbsMfbmI]



amcl (adaptive Monte Carlo localization) works by figuring out where the robot would need to be on the map in order for its laser scans to make sense. Each possible location is represented by a “particle” and particles with laser scans that do not match well are removed resulting in a group of particles representing the location of the robot in the map.

SLAM Map Building

OpenSLAM Gmapping

- https://www.openslam.org/gmapping.html
- http://wiki.ros.org/gmapping

A. SLAM Map Building with TurtleBot
    - http://wiki.ros.org/turtlebot_navigation/Tutorials/indigo/Build%20a%20map%20with%20SLAM

TurtleBot2 in Gazebo | Simulation

TurtleBot in Gazebo (Indigo)

- http://wiki.ros.org/turtlebot_simulator

Installation

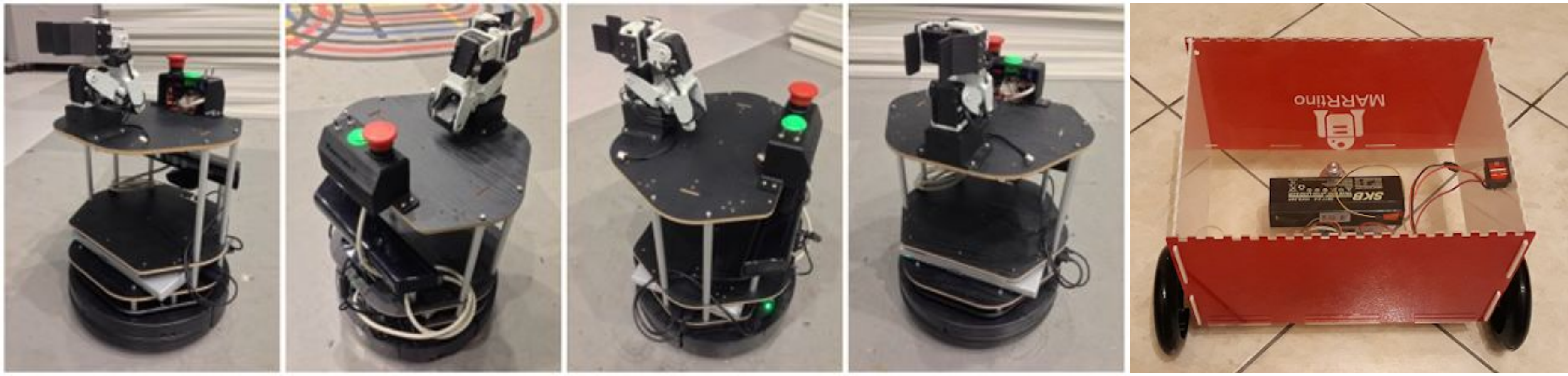
- $ sudo apt-get install ros-kinetic-turtlebot-simulator

Bring up

- $ roslaunch turtlebot_gazebo turtlebot_world.launch

Navigation: SLAM Map Building | Simulation

1. Launch Gmapping for map building
   - $ roslaunch turtlebot_gazebo gmapping_demo.launch
2. Use RViz for mapping visualization
   - $ roslaunch turtlebot_rviz_launchers view_navigation.launch
   - In rviz, Global Map > Costmap > Topic > /map
   - In rviz, Local Map > Costmap > Topic > /map
3. Use TurtleBot teleop to scan around for mapping
   - $ roslaunch turtlebot_teleop keyboard_teleop.launch
4. Save map after scanning
   - $ rosrun map_server map_saver -f /home/<username>/my_map



RoboCup@Home Education
ONLINE CHALLENGE 2020
Online Classroom Open Platform

Web: https://www.robocupathomeedu.org/challenges/robocuphome-education-online-challenge-2020
Online Classroom: https://www.robocupathomeedu.org/learn/online-classroom/online-challenge-2020
Online Entry Form: https://forms.gle/UBREeC1xTCVQ9wr78
Online Entry Form (backup): https://www.wjx.cn/jq/72082120.aspx
Contact: oc@robocupathomeedu.org

RoboCup@Home EDUCATION MathWorks® JUPITER ROBOT