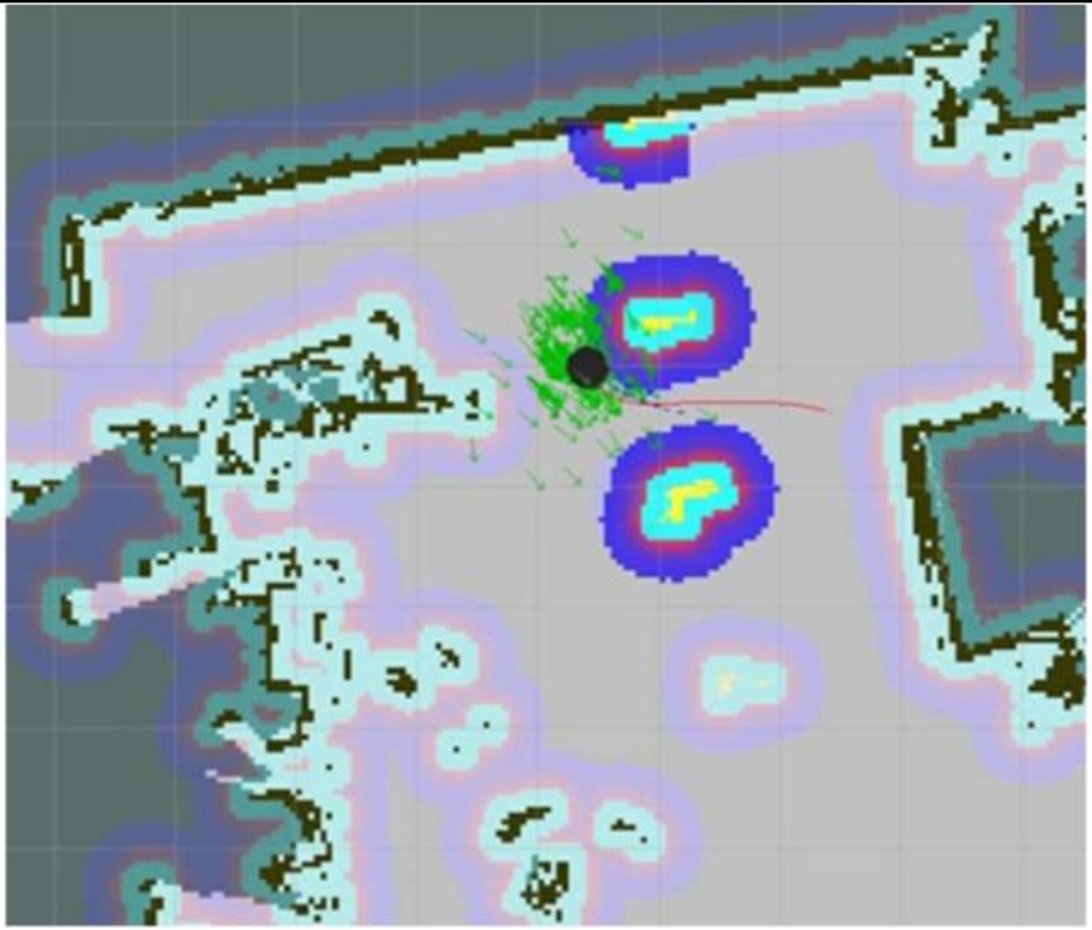
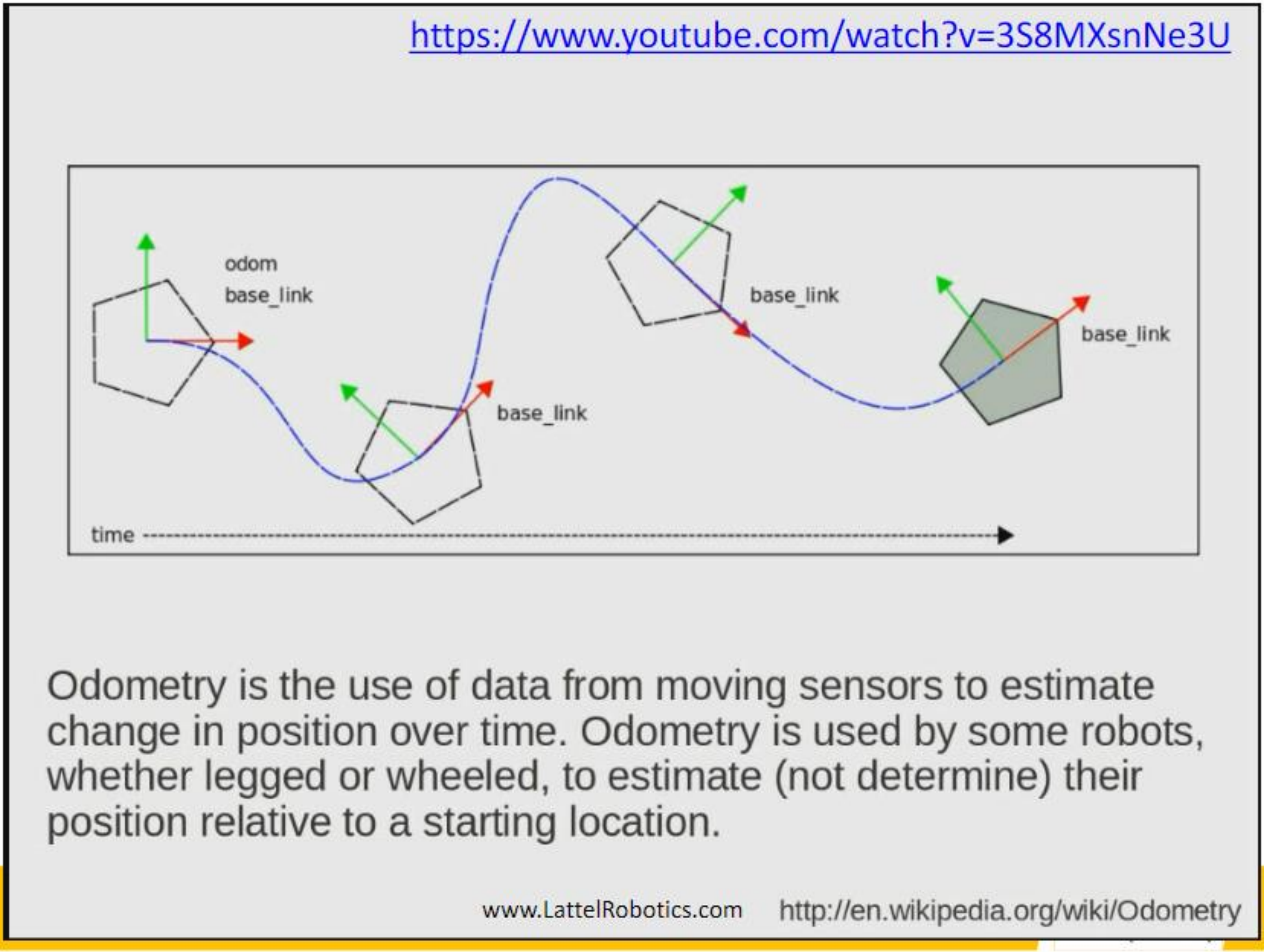


SLAM MAP BUILDING

Robot Odometry and Localization
TurtleBot Odometry

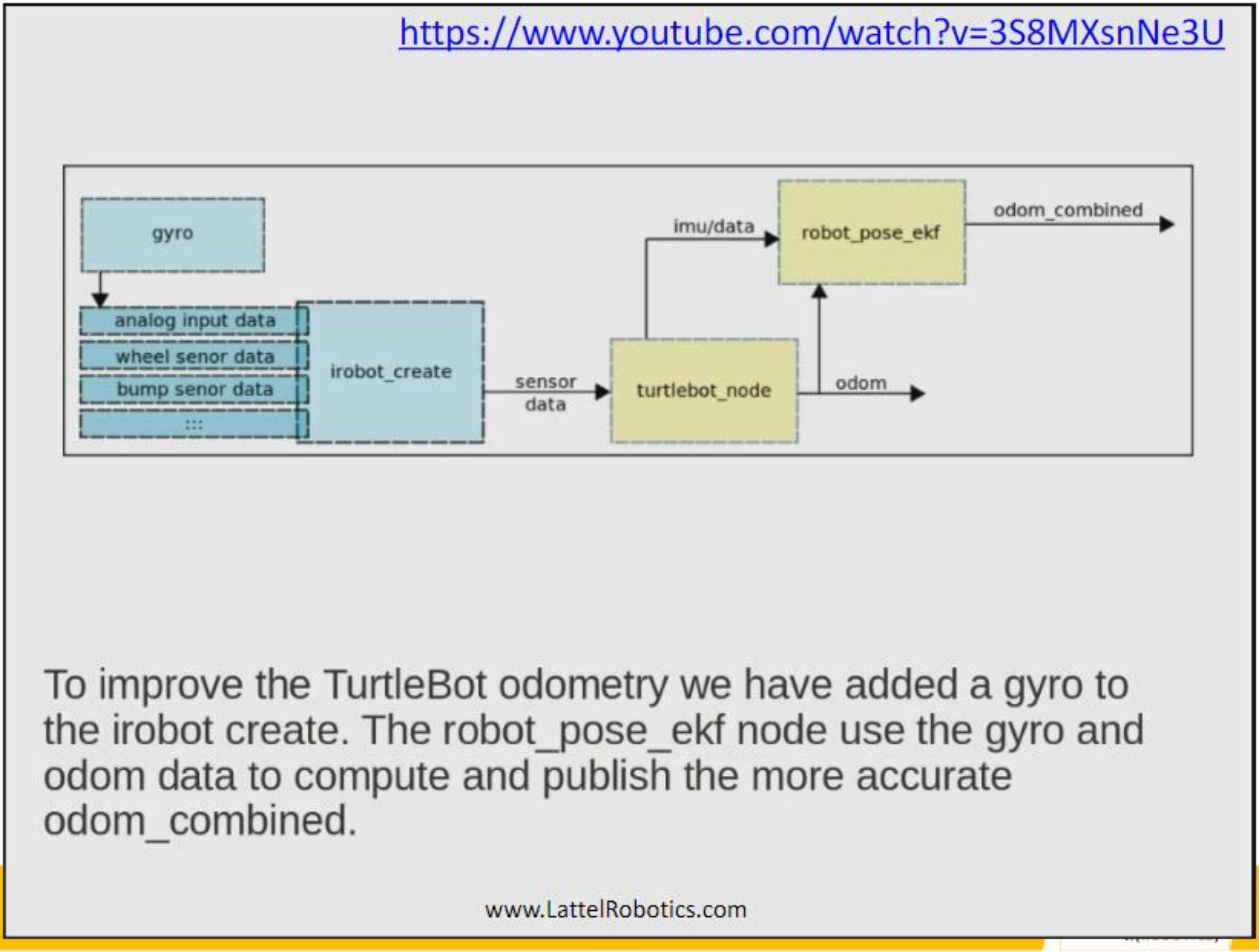


https://www.youtube.com/watch?v=3S8MXsnNe3U

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

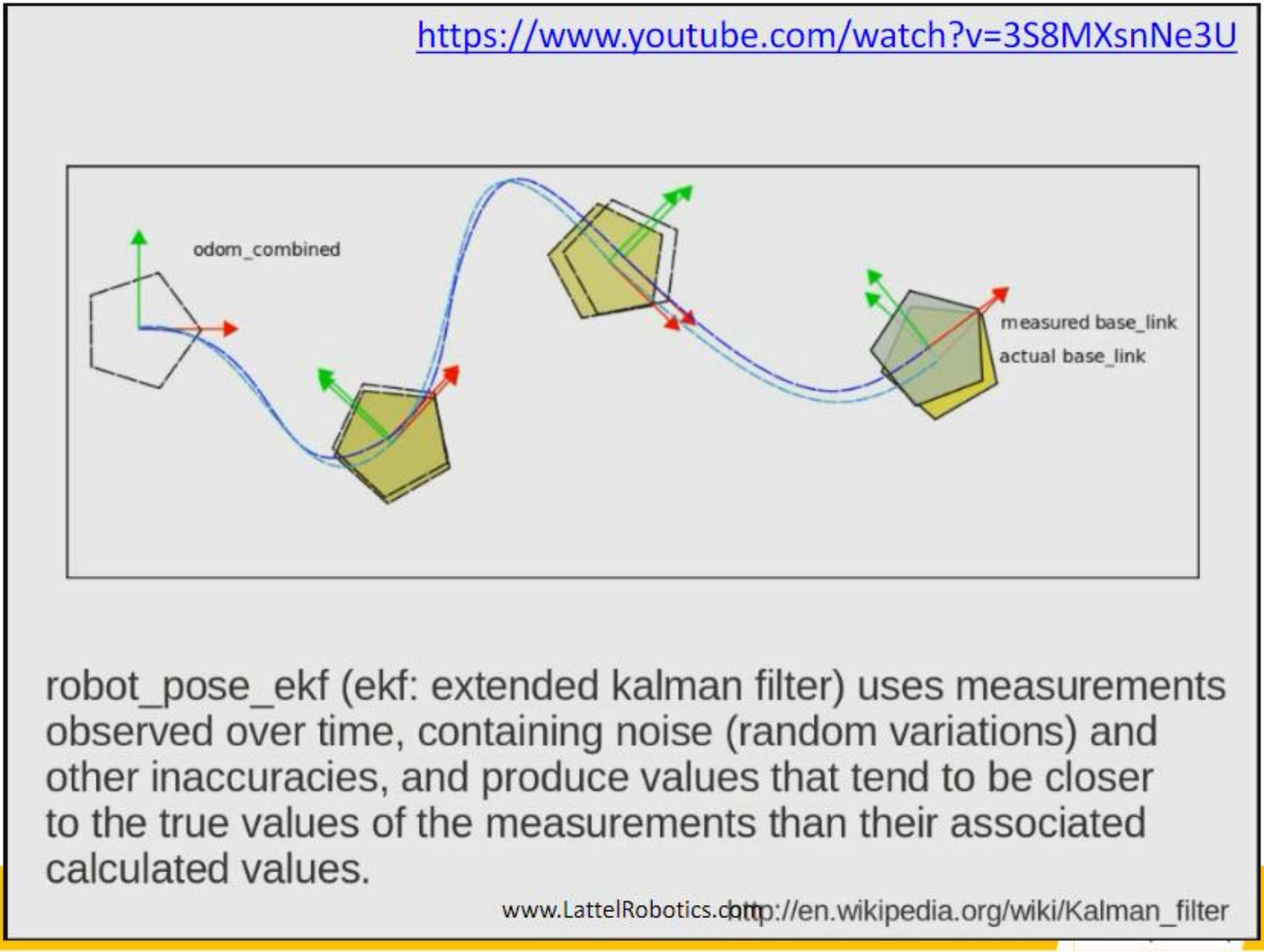
Robot Odometry and Localization
TurtleBot Odometry



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

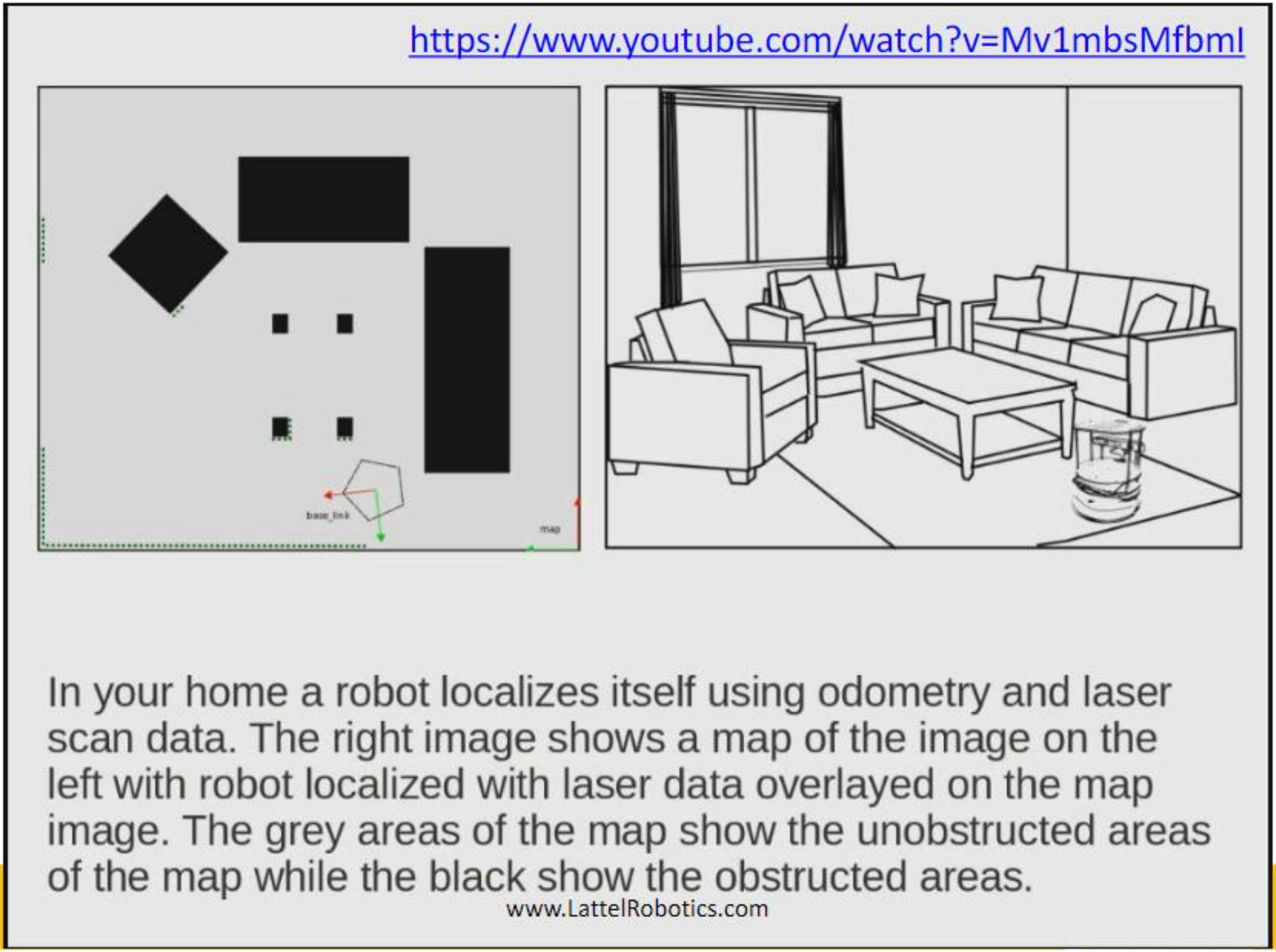


https://www.youtube.com/watch?v=3S8MXsnNe3U

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

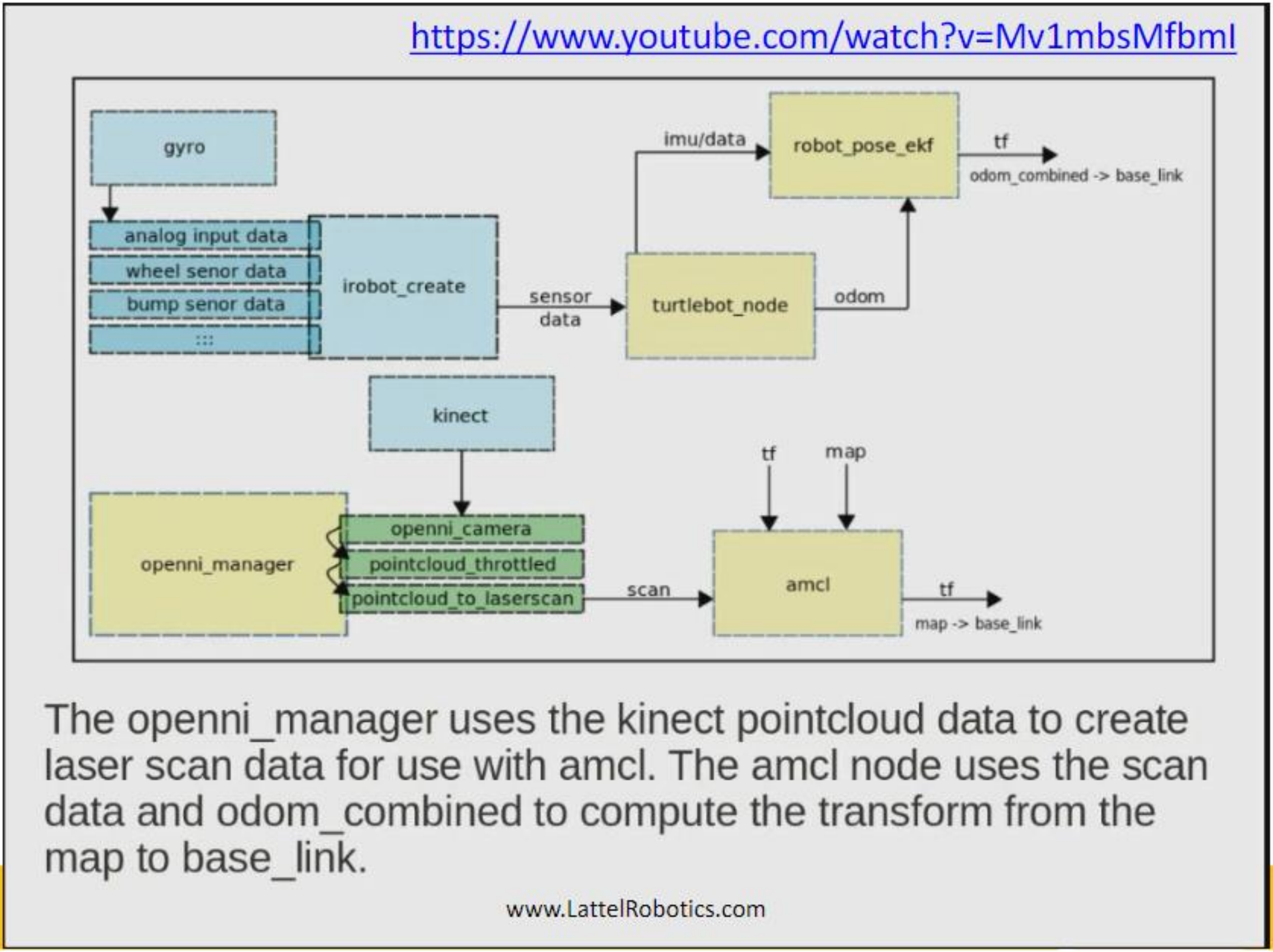
Robot Odometry and Localization
TurtleBot Localization



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.

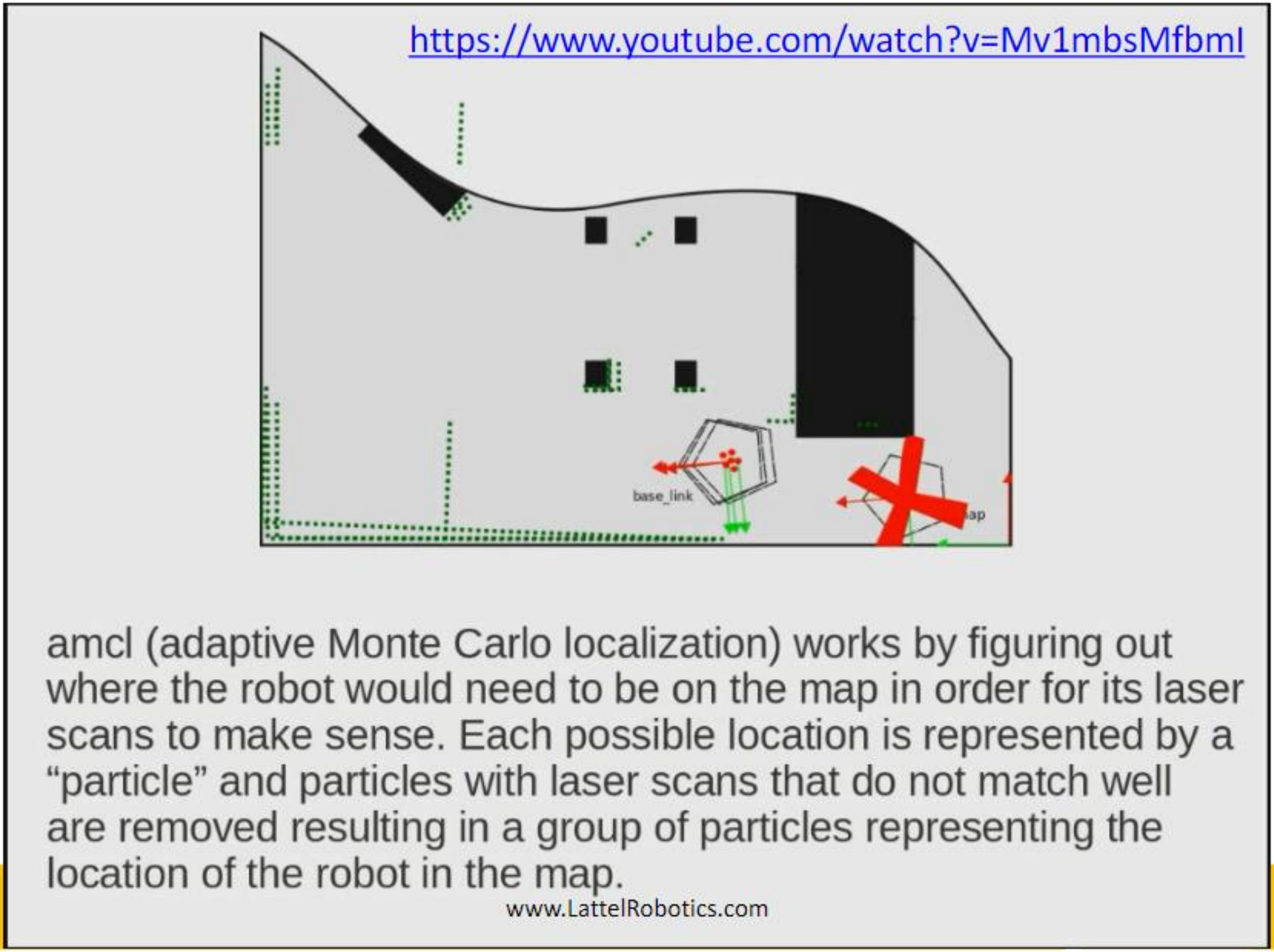
Robot Odometry and Localization
TurtleBot Localization



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



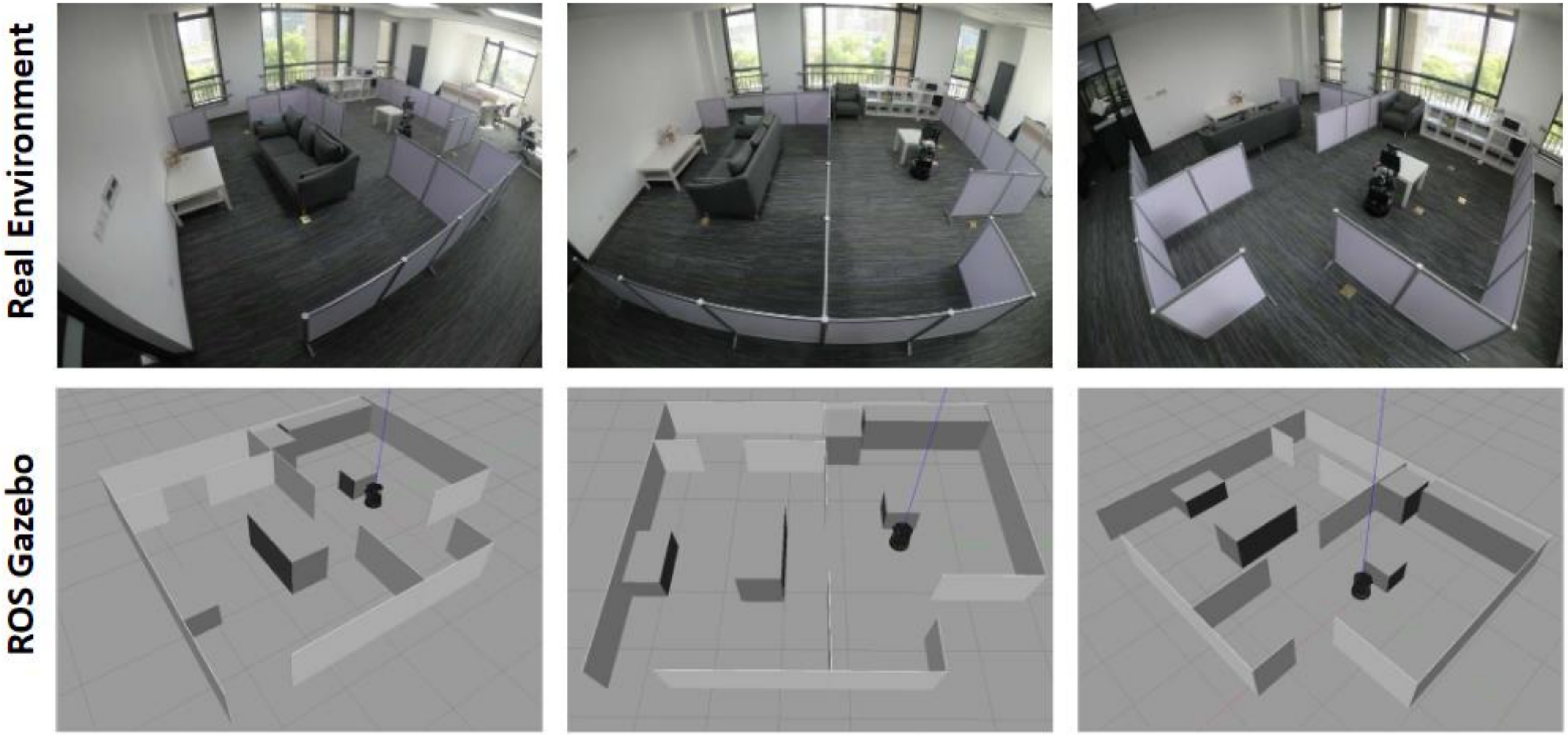
https://www.youtube.com/watch?v=Mv1mbsMfbmI

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.

Realistic Simulation Environment for Service Robot Development

RoboCup@Home EDUCATION

JUPITER ROBOT



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

SLAM Map Building [Robot]

1. Bring up
   - $ roslaunch jupiterobot_bringup **jupiterobot_bringup.launch**
2. Launch Gmapping for map building
   - [RGB-D] $ roslaunch jupiterobot_navigation **gmapping_demo.launch**
   - [Lidar] $ roslaunch jupiterobot_navigation **rplidar_gmapping_demo.launch**
3. Use RViz for mapping visualization
   - $ roslaunch turtlebot_rviz_launchers **view_navigation.launch**
4. Use teleop to scan around for mapping
   - [Keyboard] $ roslaunch turtlebot_teleop **keyboard_teleop.launch**
   - [Gamepad] $ roslaunch jupiterobot_teleop_move **joy_move.launch**
5. Save map after scanning
   - $ rosrun map_server map_saver -f /home/mustar/catkin_ws/maps/test1

Jupiter Robot in Gazebo [Simulation]

- Launch robot in virtual world

  -
    ```
    $ roslaunch jupiterobot_gazebo jupiterobot_world.launch world_file:=/home/mustar/catkin_ws/worlds/Jupiter_Robot_Office.world
    ```

- Simulation model parameters
  - stacks: **h** (hexagon plates), **c** (circular plates) | default – **h**
  - lasers: **n** (none), **r** (rplidar), **h** (hokuyo) | default – **r**
  - arms: **n** (none), **5** (5 DOF arm), **7** (7 DOF arm) | default – **5**
  - heads: **n** (none), **1** (1 DOF head), **2** (2 DOF head) | default – **1**

SLAM Map Building [Simulation]

1. Launch Gmapping for map building

   -
     ```
     $ roslaunch jupiterobot_gazebo rplidar_gazebo_gmapping_demo.launch
     ```
2. Use RViz for mapping visualization

   -
     ```
     $ roslaunch turtlebot_rviz_launchers view_navigation.launch
     ```
3. Use teleop to scan around for mapping

   -
     ```
     $ roslaunch turtlebot_teleop keyboard_teleop.launch
     ```
4. Save map after scanning

   -
     ```
     $ rosrun map_server map_saver -f /home/mustar/catkin_ws/maps/test2
     ```