Autonomous Robotics Motion Planning

Benchun Zhou

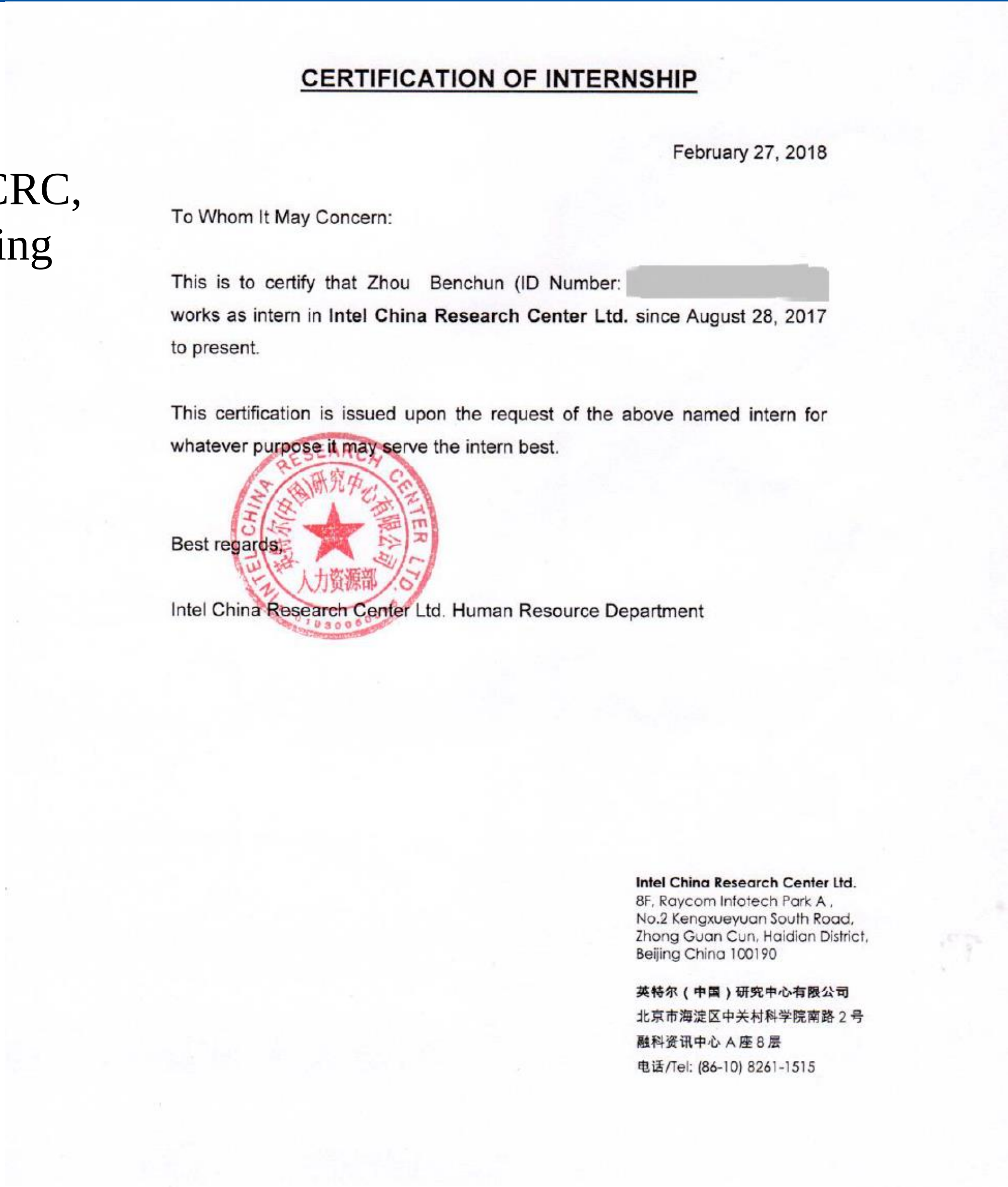
School of Automation Science and Electronic Engineering
Beihang University

CONTENTS

1. Introduction

- **Introduction**

This document presents the internship at Intel China Research Center (ICRC, Beijing), in the work, we are going to create a real autonomous robot using ROS



CERTIFICATION OF INTERNSHIP

February 27, 2018

To Whom It May Concern:

This is to certify that Zhou Benchun (ID Number: [redacted] works as intern in **Intel China Research Center Ltd.** since August 28, 2017 to present.

This certification is issued upon the request of the above named intern for whatever purpose it may serve the intern best.

INTEL CHINA RESEARCH CENTER LTD.
英特尔(中国)研究中心有限公司
人力资源部

Best regards,

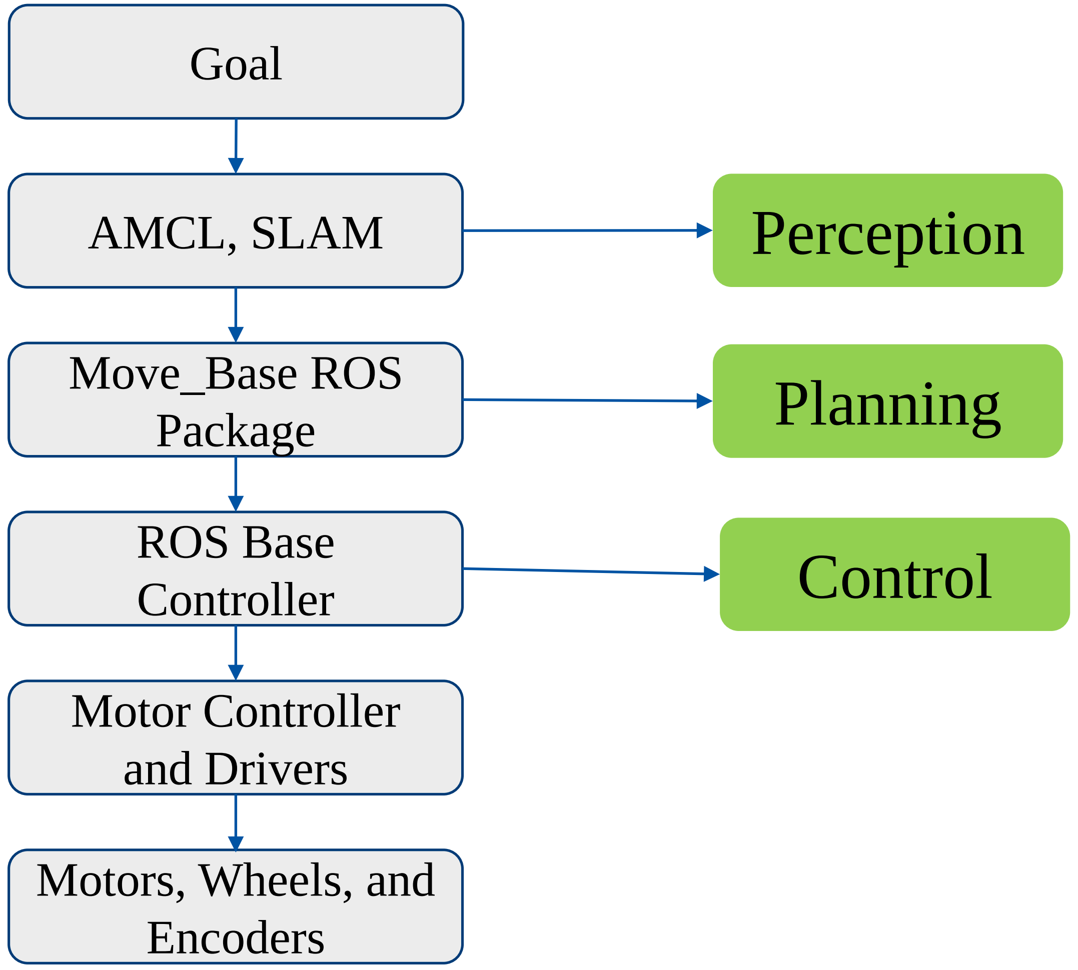
Intel China Research Center Ltd. Human Resource Department

Intel China Research Center Ltd.
8F, Raycom Infotech Park A,
No.2 Kengxueyuan South Road,
Zhong Guan Cun, Haidian District,
Beijing China 100190

英特尔(中国)研究中心有限公司
北京市海淀区中关村科学院南路2号
融科资讯中心A座8层
电话/Tel: (86-10) 8261-1515

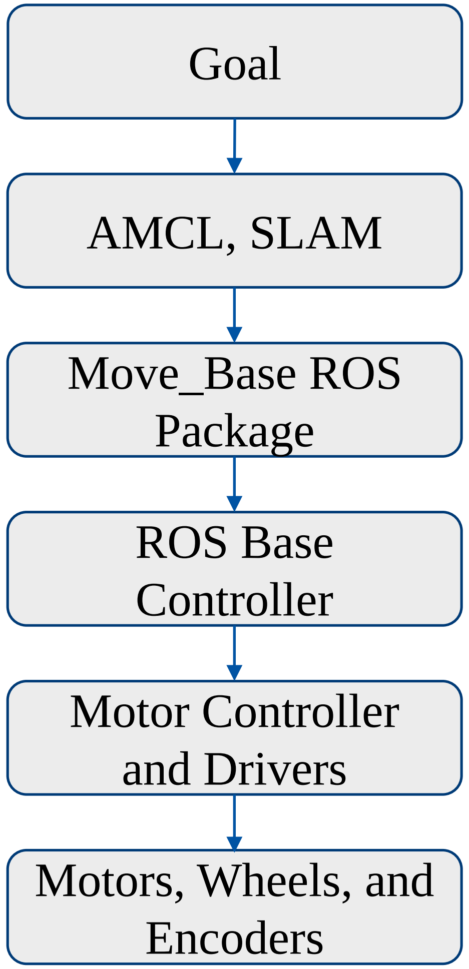
1. Introduction

- **Navigation Framework**

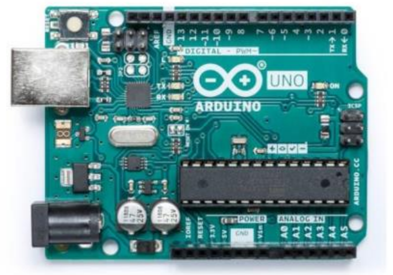


2. Hardware

- **Framework**
- hardware



Rplidar


Arduino UNO R3

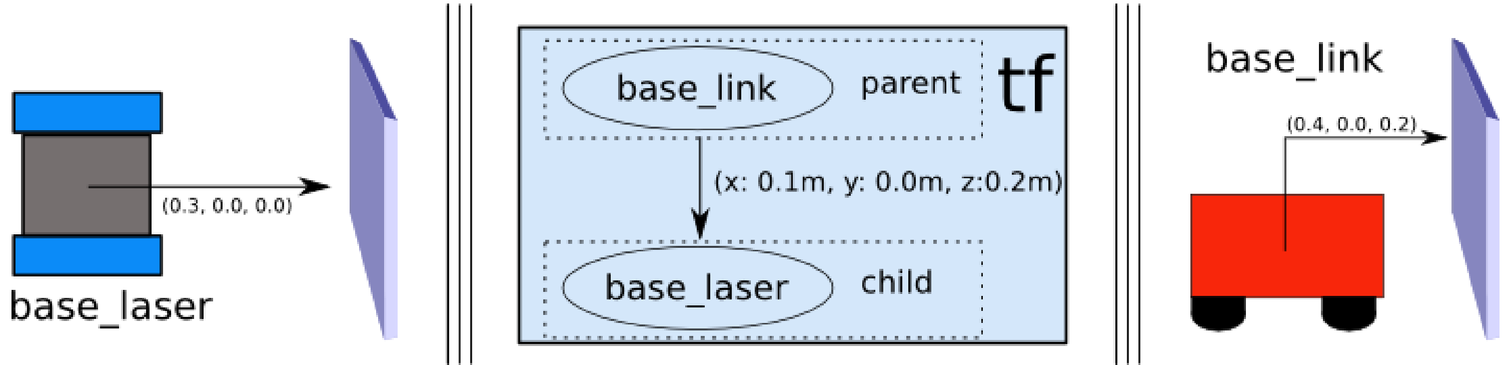
Encoder

3. Software

- **Transform Configuration**

- tf lib: translation and rotation between different coordinate frames
- Advantages: flexible, simple API, and easy to find the relationship between different frame
- Disadvantages: computing complexity, not real time, use transform tree

- **Frame: map->odom-->base_link->base_laser**
- Map: fix frame of the world
- Odom: come from wheel or computer vision, to estimate the posture of robot
- Base-link: generally, the center of robot
- Base-laser: the center of sensor



3. Software

- **Base controller**

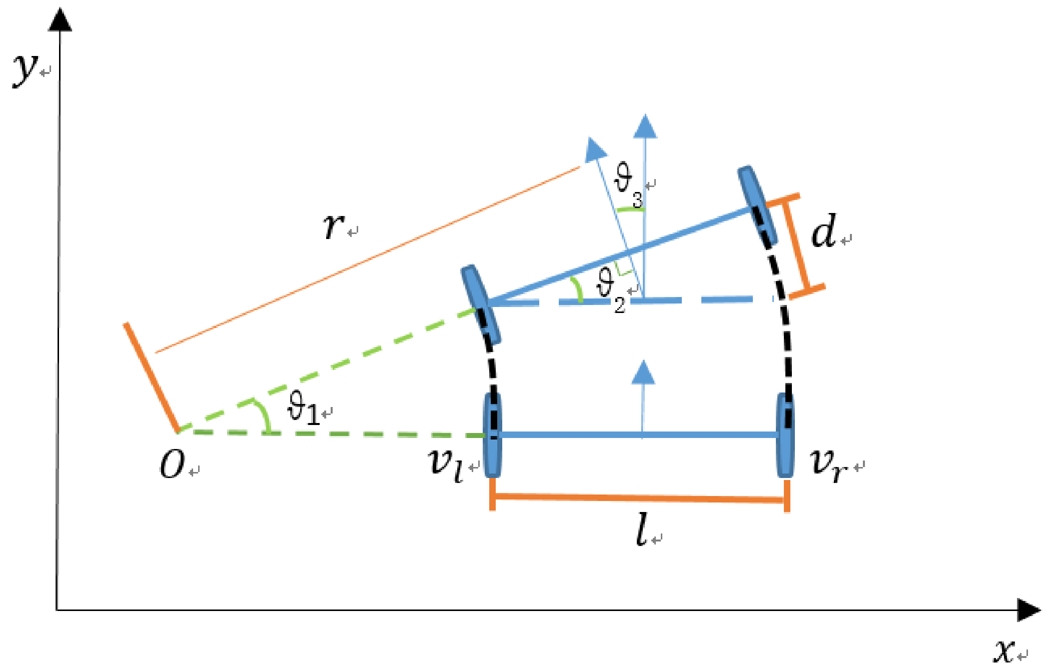
- Target1: receive command and control the wheel
  - Cmd_vel (Twist message)

Geometry_msgs/Vector3 linear
  float64 x $\rightarrow \boxed{v}$
  float64 y
  float64 z

Geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z $\rightarrow \omega$

- Kinematic model

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \longrightarrow \begin{aligned} x_{t+1} &= x_t + v\Delta t\cos\theta_t \\ y_{t+1} &= y_t + v\Delta t\sin\theta_t \\ \theta_{t+1} &= \theta_t + \omega\Delta t \end{aligned}$$



3. Software

- **Base controller**

- Target2: publish odometry information

- Linear velocity of robot

$$v = \frac{v_r + v_l}{2}$$

- Angular velocity

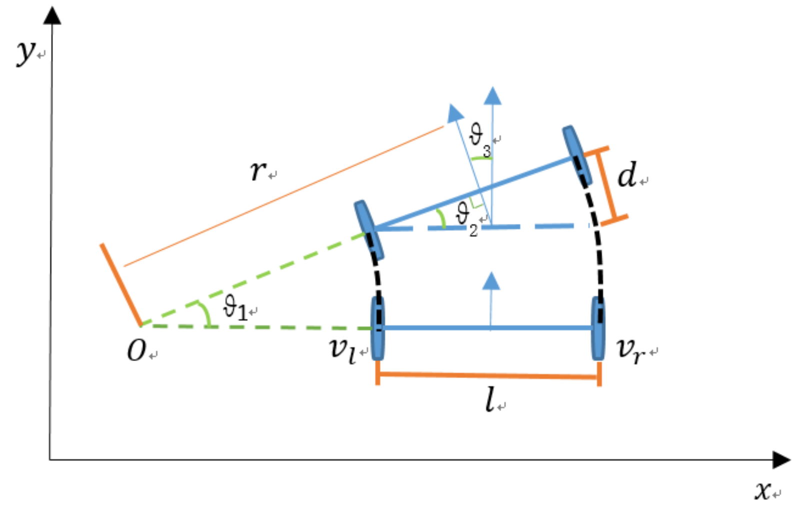
$$\omega = \frac{\theta_1}{\Delta t} = \frac{(v_r - v_l)}{l}$$

- Twist message

Geometry_msgs/Vector3 linear
float64 x $\longleftarrow \boxed{v}$
float64 y
float64 z

Geometry_msgs/Vector3 linear
float64 x
float64 y
float64 z $\longleftarrow \omega$

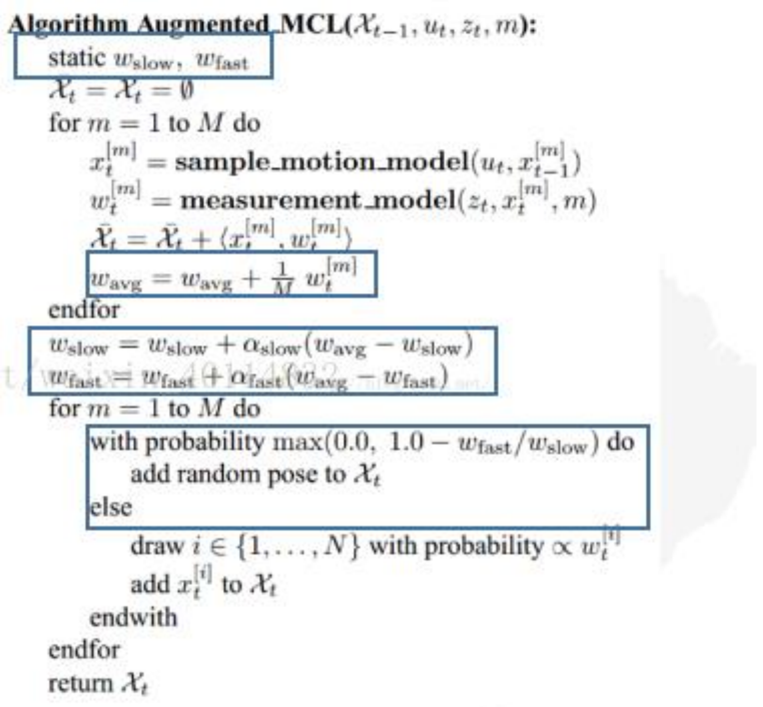
- Publish: ros_arduino_bridge



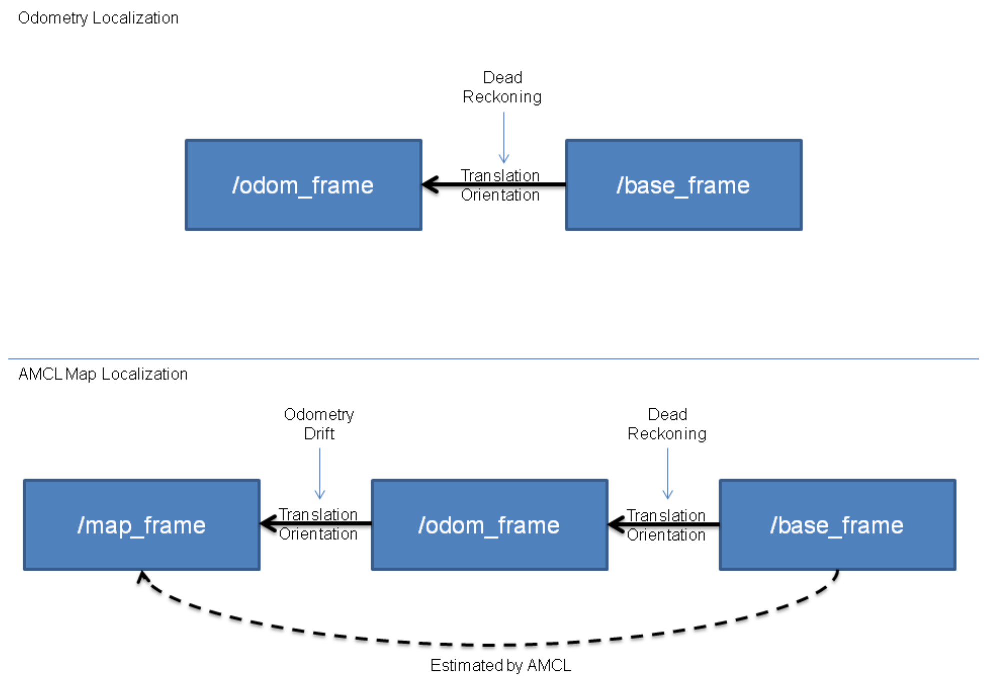
3. Software

- **SLAM**
- AMCL (Adaptive Monte Carlo Localization)

amcl is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map.



```
Algorithm Augmented_MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
    static $w_{\text{slow}}, w_{\text{fast}}$
    $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
    for $m = 1$ to $M$ do
        $x_t^{[m]} =$ sample_motion_model($u_t, x_{t-1}^{[m]}$)
        $w_t^{[m]} =$ measurement_model($z_t, x_t^{[m]}, m$)
        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
        $w_{\text{avg}} = w_{\text{avg}} + \frac{1}{M} w_t^{[m]}$
    endfor
    $w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$
    $w_{\text{fast}} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$
    for $m = 1$ to $M$ do
        with probability $\max(0.0,\ 1.0 - w_{\text{fast}}/w_{\text{slow}})$ do
            add random pose to $\mathcal{X}_t$
        else
            draw $i \in \{1, \ldots, N\}$ with probability $\propto w_t^{[i]}$
            add $x_t^{[i]}$ to $\mathcal{X}_t$
        endwith
    endfor
    return $\mathcal{X}_t$
```

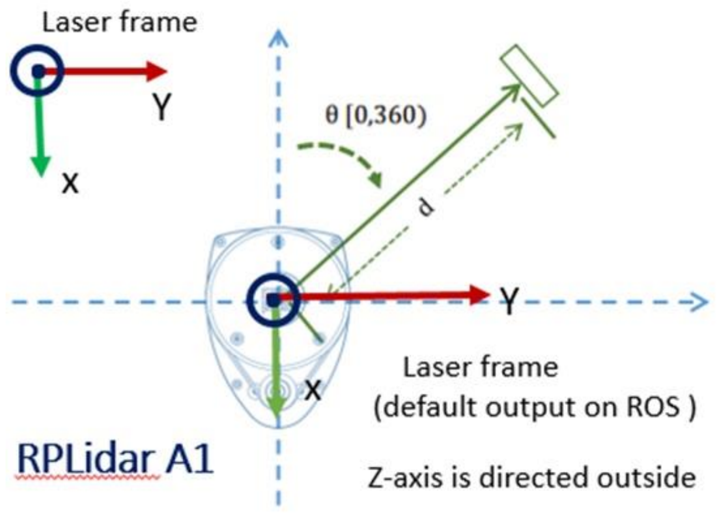


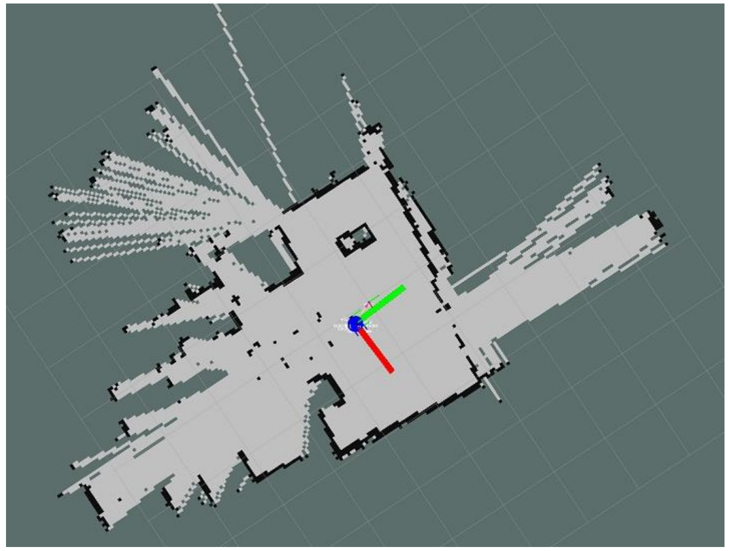
3. Software

- **SLAM**

- Gmapping

The gmapping package provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called slam_gmapping. Using slam_gmapping, you can create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

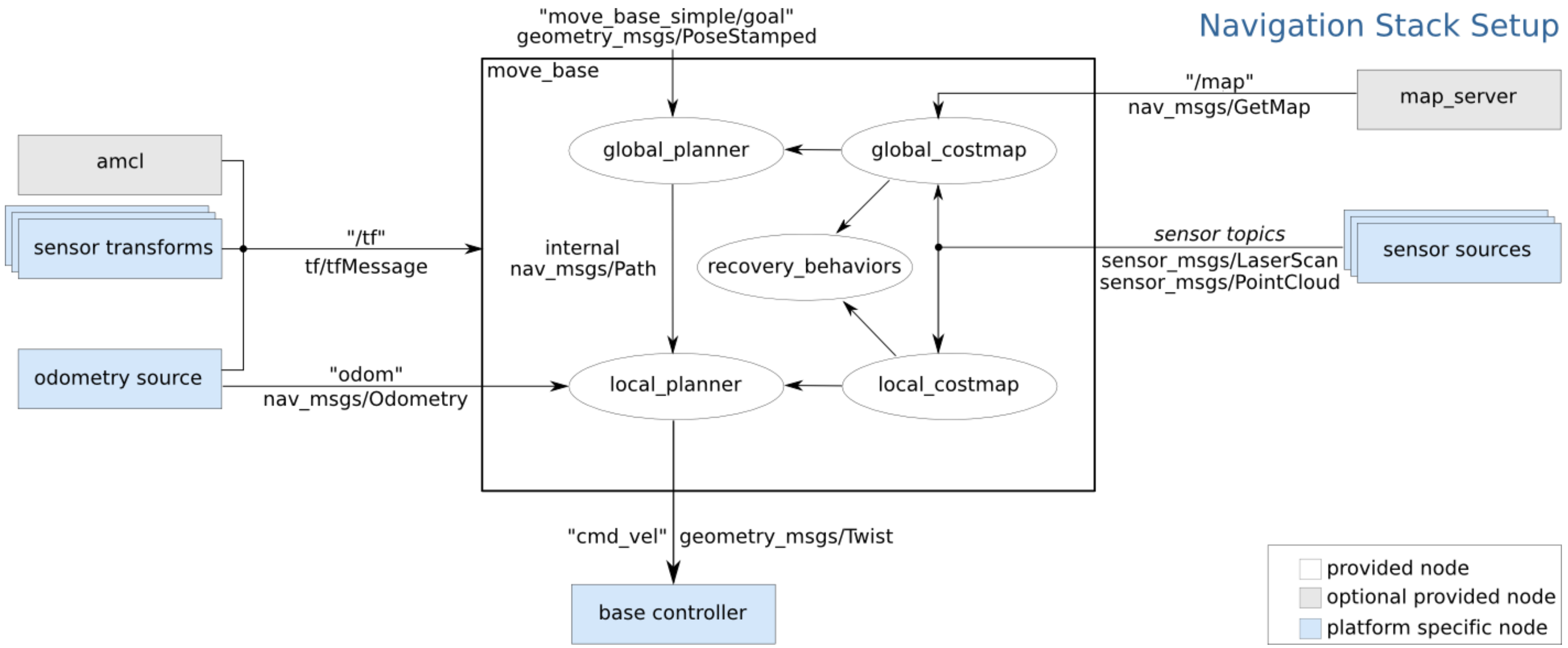




3. Software

- **Move_base**
- Function

The move_base package provides an implementation of an action that, given a goal in the world, will attempt to reach it with a mobile base. The move_base node links together a global and local planner to accomplish its global navigation task



3. Software

- **Move_base**
- Structure

  global_planner： receive global_costmap, employ A*/D* to planning a path from initial position to target position, provide reference to local planner

  local_planner： receive global_costmap, employ DWA (Dynamic Window Approach) to publish cmd_vel to base_controller, considering reference trajectory and obstacle avoidance

  global_costmap： generate global cost map as occupancy grid map to global planner

  local_costmap： generate local cost map as occupancy grid map to local planner

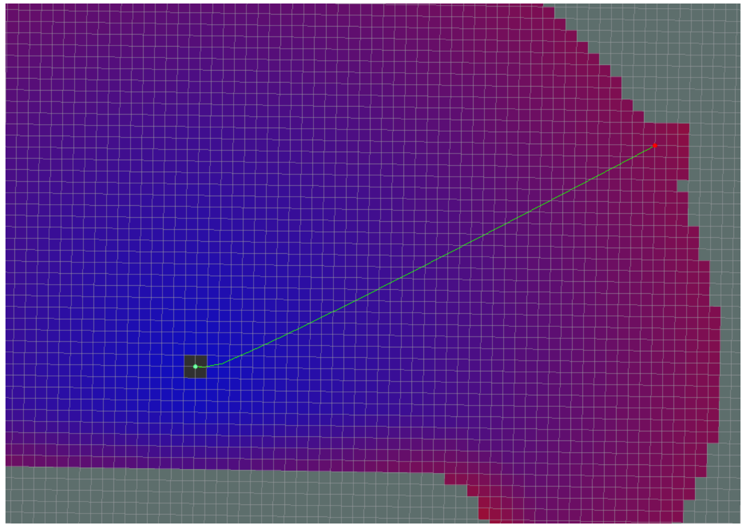
- Other methods

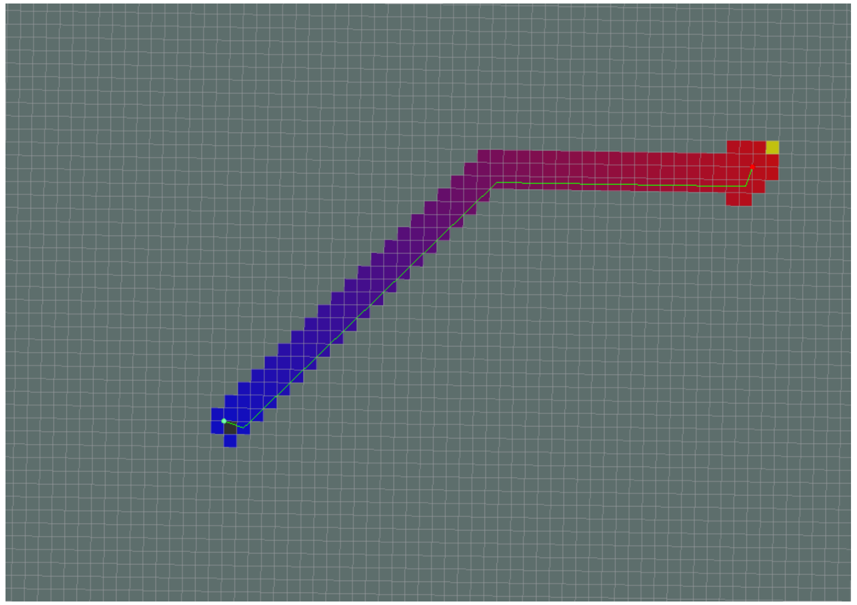
  RRT (Rapid Random-exploring Tree)

3. Software

- **Move_base**
- Global planner

This package provides an implementation of a fast, interpolated global planner for navigation. This class adheres to the nav_core::BaseGlobalPlanner interface specified in the nav_core package. It was built as a more flexible replacement to navfn,


dijsktra

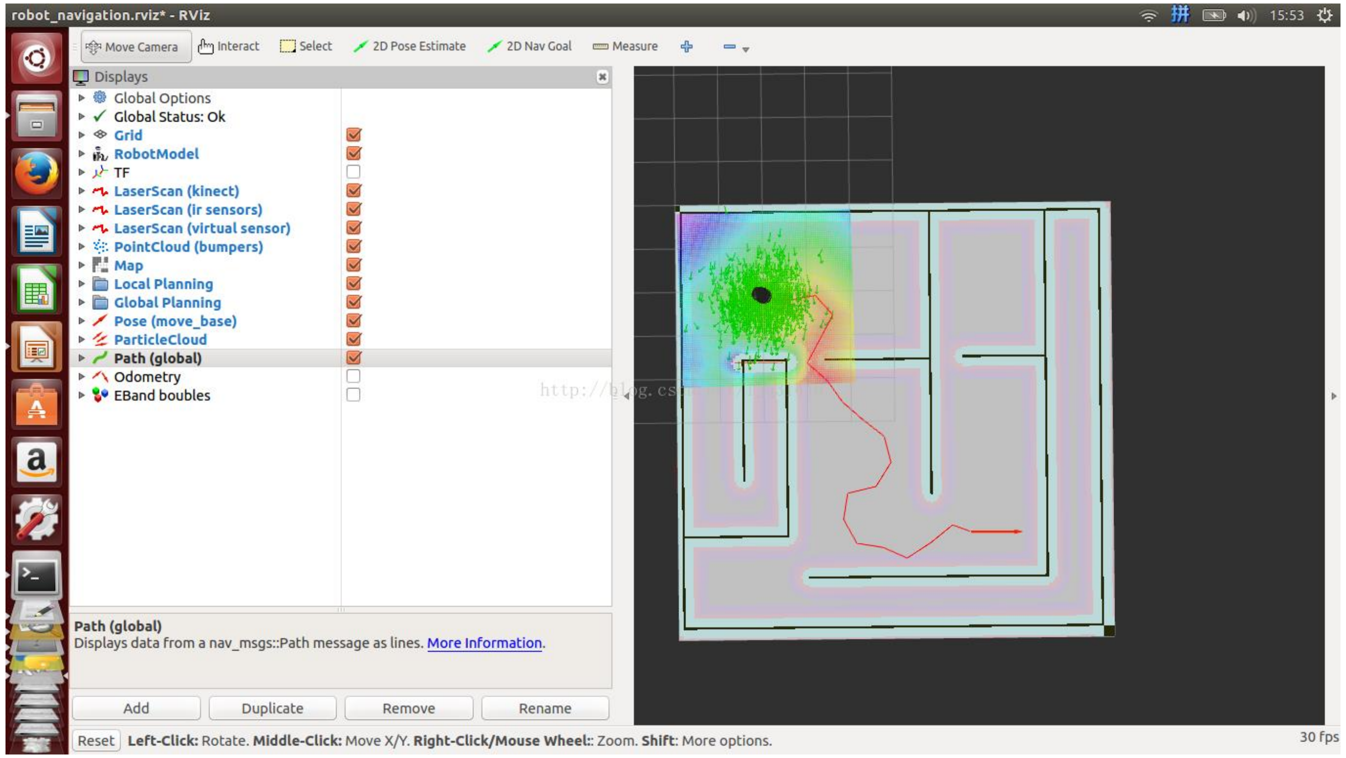

A*

3. Software

BEIHANG UNIVERSITY
——BEIJING, CHINA——

- **Move_base**
- Global planner

This package provides an implementation of a fast, interpolated global planner for navigation. This class adheres to the nav_core::BaseGlobalPlanner interface specified in the nav_core package. It was built as a more flexible replacement to navfn,



Path (global)
Displays data from a nav_msgs::Path message as lines. More Information.

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.

RRT

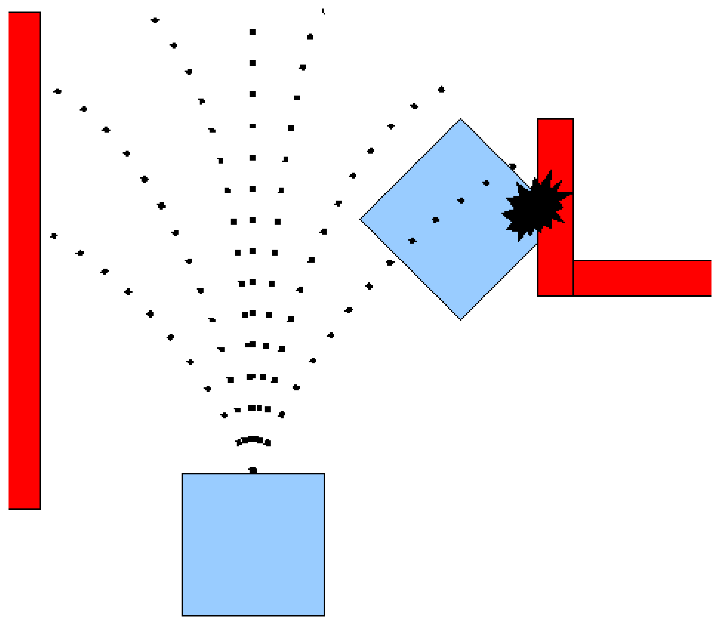
3. Software

- **Move_base**
- Local planner (DWA)

Using a map, the planner creates a kinematic trajectory for the robot to get from a start to a goal location. Along the way, the planner creates, at least locally around the robot, a value function, represented as a grid map. This value function encodes the costs of traversing through the grid cells. The controller's job is to use this value function to determine dx,dy,dtheta velocities to send to the robot.

The basic idea of the Dynamic Window Approach (DWA) algorithm is as follows:

- ➢ Discretely sample in the robot's control space (dx,dy,dtheta)
- ➢ For each sampled velocity, perform forward simulation from the robot's current state to predict what would happen if the sampled velocity were applied for some (short) period of time.
- ➢ Evaluate (score) each trajectory resulting from the forward simulation, using a metric that incorporates characteristics such as: proximity to obstacles, proximity to the goal, proximity to the global path, and speed. Discard illegal trajectories (those that collide with obstacles).
- ➢ Pick the highest-scoring trajectory and send the associated velocity to the mobile base.
- ➢ Rinse and repeat.



3. Software

- **Move_base**
- Local planner (DWA)

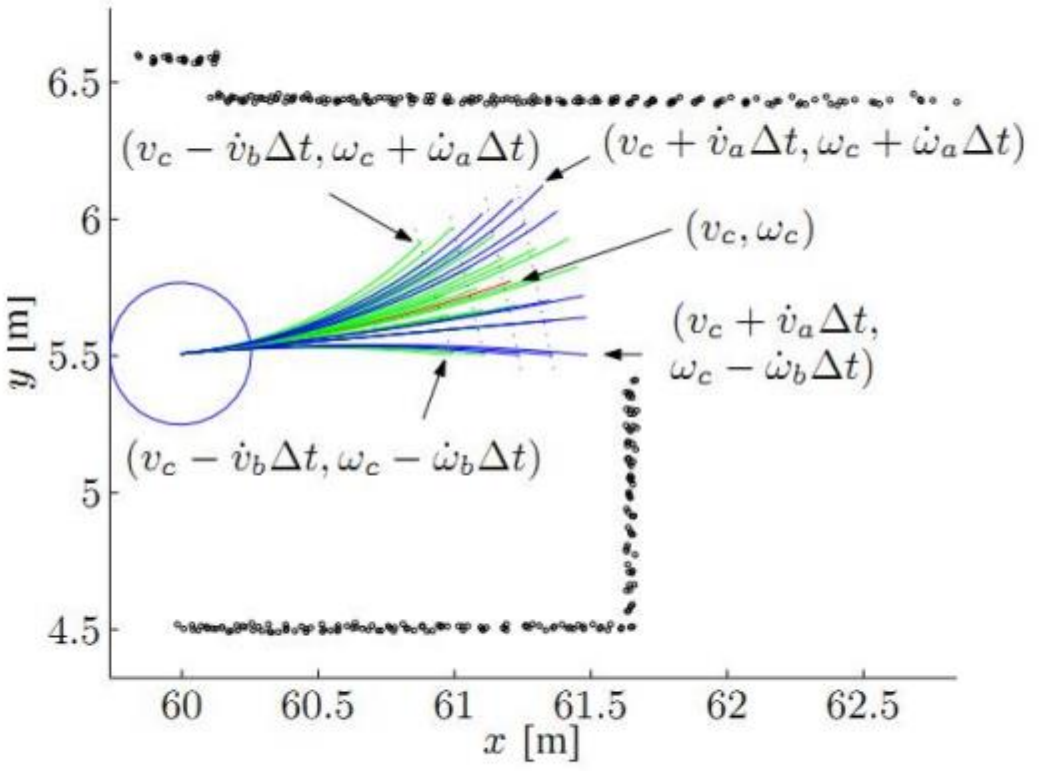
Velocity constraints
$$V_m = \{ v \in [v_{\min}, v_{\max}], \omega \in [\omega_{\min}, \omega_{\max}] \}$$

Acceleration constraints
$$V_d = \left\{ (v,\omega) \middle| \begin{array}{l} v \in [v_c - \dot{v}_b \Delta t, v_c + \dot{v}_a \Delta t] \\ \omega \in [\omega_c - \dot{\omega}_b \Delta t, v_c + \dot{\omega}_a \Delta t] \end{array} \right\}$$

Safety constraints
$$V_a = \left\{ (v,\omega) \middle| \begin{array}{l} v \le \sqrt{2 \times dist(v,\omega) \times \dot{v}_b} \\ \omega \le \sqrt{2 \times dist(v,\omega) \times \dot{\omega}_b} \end{array} \right\}$$

Cost function

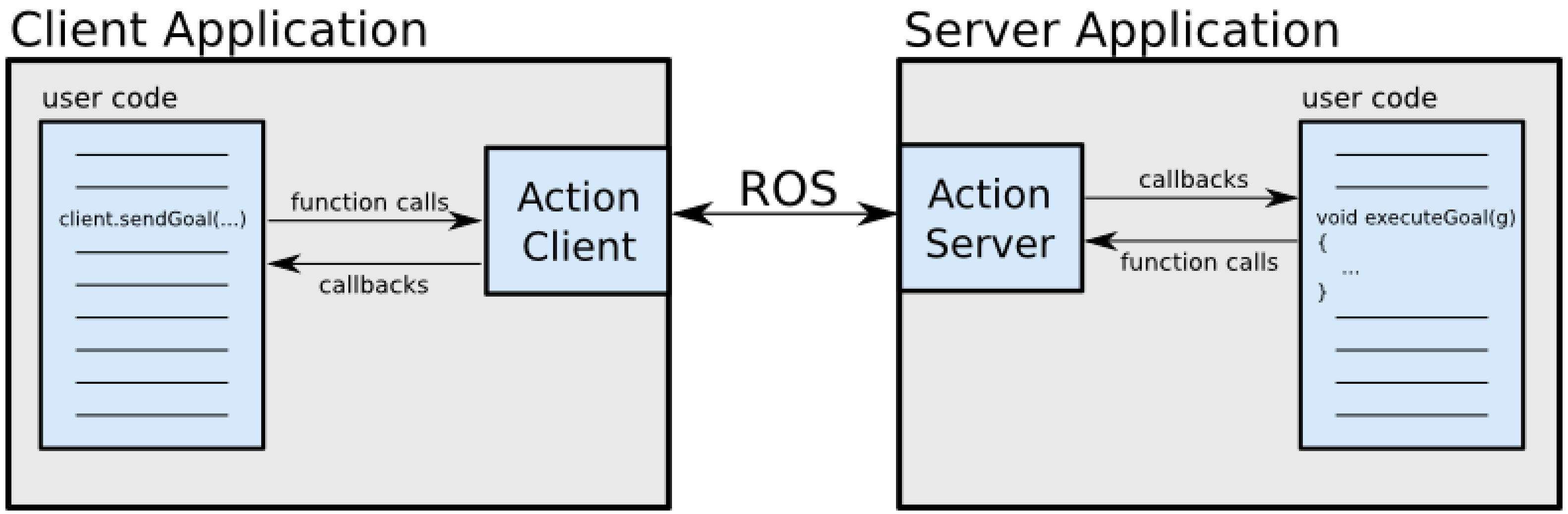
$$G(v,\omega) = \sigma\left(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega)\right)$$



3. Software

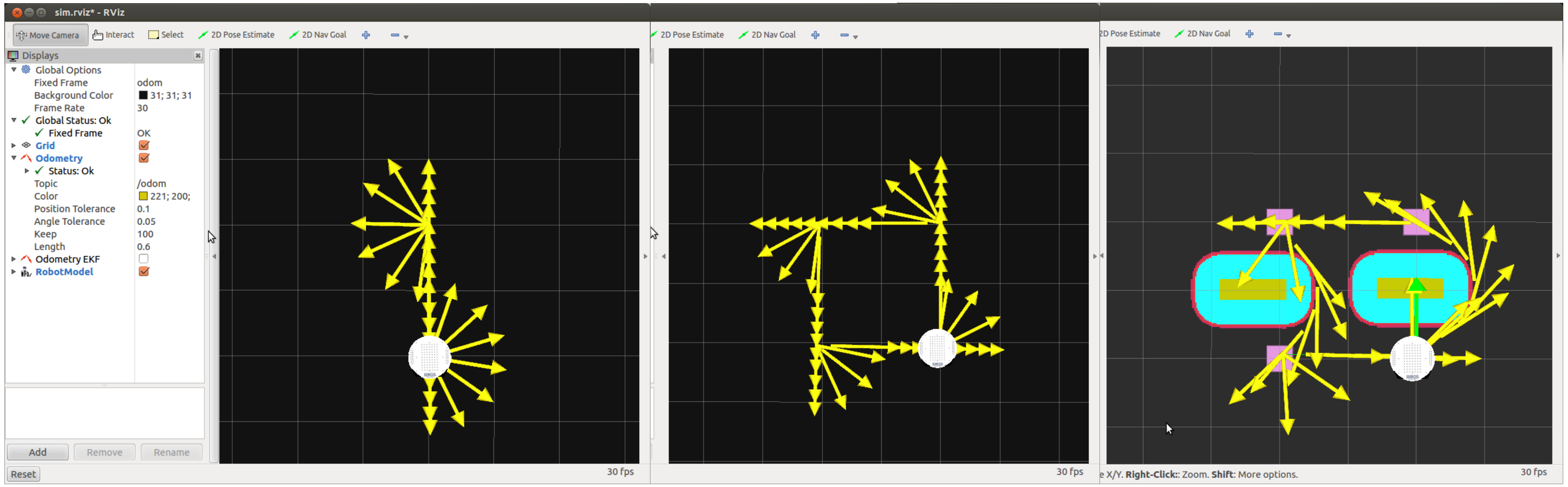
- **Goal Send**
- Actionlib

The actionlib stack provides a standardized interface for interfacing with preemptable tasks. Examples of this include moving the base to a target location, performing a laser scan and returning the resulting point cloud, detecting the handle of a door, etc.



4. Simulation

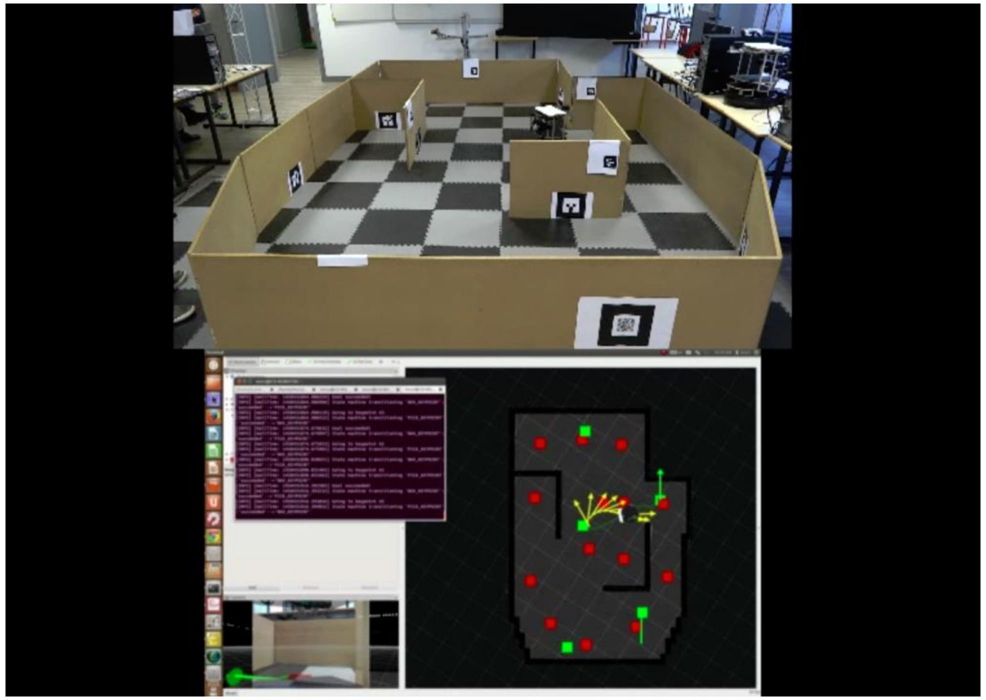
- **Feedback test**
- Line test, circle test and path planning test



```
Global Options
  Fixed Frame              odom
  Background Color         31; 31; 31
  Frame Rate               30
✓ Global Status: Ok
  ✓ Fixed Frame            OK
Grid                       ✓
Odometry                   ✓
  ✓ Status: Ok
  Topic                    /odom
  Color                    221; 200;
  Position Tolerance       0.1
  Angle Tolerance          0.05
  Keep                     100
  Length                   0.6
Odometry EKF               ☐
RobotModel                 ✓
```

5. Practice

- **Real robot test**



THANKS YOU!