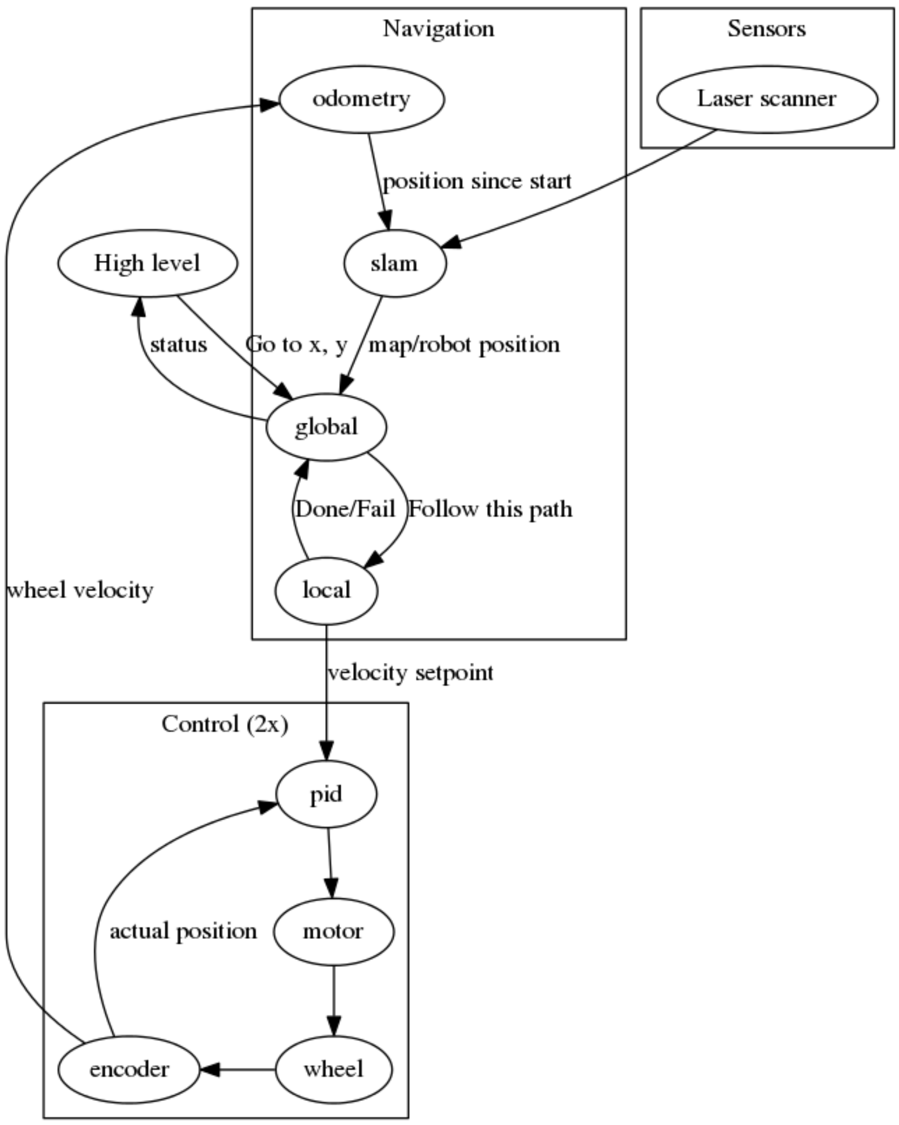
1 Introduction

This document describes how the problem of 2d-navigation was solved on a real robot. The goal of navigation is to safely move a robot from a to b. The robot used was moving base, the x80sv.

The task of navigation can be split up into the following tasks:

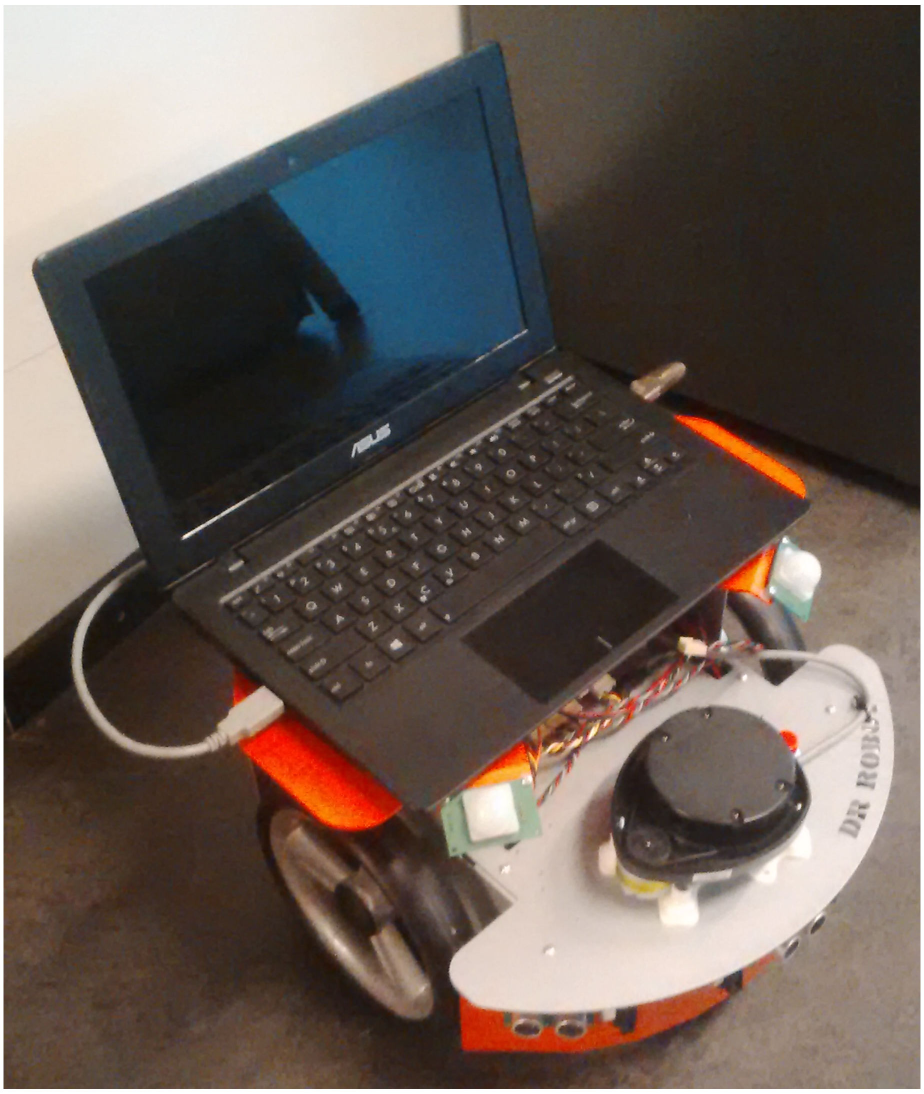
- Robot control. This task is concerned with controlling the seperate wheels of the robot. Typically this is done using PID control or something alike.
- Odometry calculation. By keeping track of wheel motions, the robot position can be calculated. This method is subjective to drift.
- Position and mapping, also known as SLAM, is the task of determining location of the robot in the world, and at the same time reconstructing the world.
- Global planning is the task of determining a path through a known map. This requires a search like A* or something like that.
- Local planning takes as input the global path and generated the appropriate motion commands for the robot control. It is some sort of setpoint generator. This layer is also responsible for obstacles. When an obstacle is observed, the path may be adjusted.



The rest of this document describes all the tasks listed above as applied to the x80sv using the robot operating system (ROS).

2 Robot setup

The robot used in this setup is the x80sv of drrobot. This robot has three wheels, of which two are controlled. Also standard sensors are range sensors, infrared and ultrasonic. The robot is extended with a laser range sensor a laptop with ROS installed. The LRS and the controllerboard of the x80sv are connected via usb-serial cables. The controllerboard of the x80sv is provided with the robot.

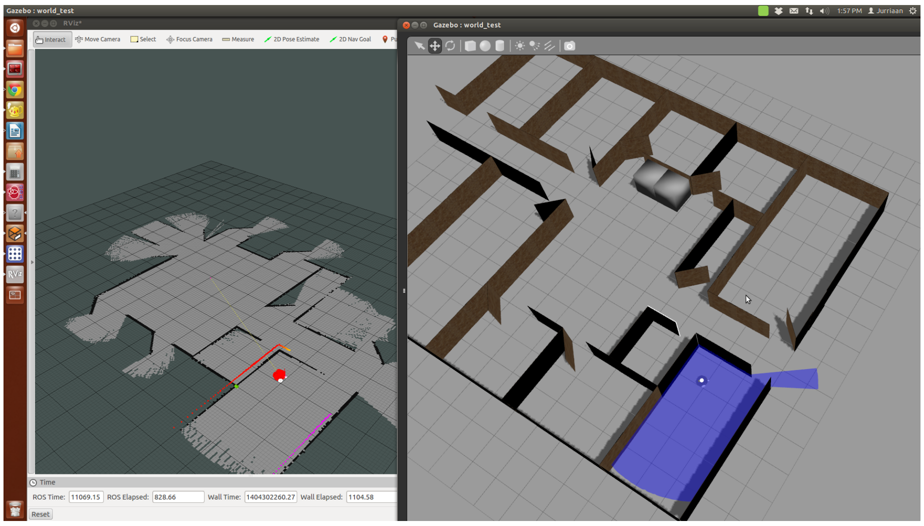


3 Installation

To use the robot, install ROS indigo on ubuntu 14.04 on the robot laptop.
Download the x80sv software from the github repository:
https://github.com/saxionled/x80sv
And follow instruction in the readme.

4 Simulation

Instead of trying to run everything on the real robot, a simulated version of the x80sv was created. This was done using the gazebo simulator.



ROS Time: 11069.15 ROS Elapsed: 828.66 Wall Time: 1404302260.27 Wall Elapsed: 1104.58

5 Control

The velocity control is done by the controllerboard of the robot. The commands to the controllerboard are wheel velocities in encoder ticks. A conversion from linear and angular velocities to wheel velocities is required.

$$v_{left} = (2 * v_{linear} - v_{angular} * wheelDistance)/(2 * wheelRadius) \quad (1)$$

$$v_{right} = (v_{angular} * wheelDistance + 2 * v_{linear})/(2 * wheelRadius) \quad (2)$$

In the next step, the wheel velocities are translated into encoder velocities.

$$leftWheelCmd = -motorDir * v_{left} * encoderOneCircleCnt/(2\pi) \quad (3)$$

$$rightWheelCmd = motorDir * v_{right} * encoderOneCircleCnt/(2\pi) \quad (4)$$

6 Odometry

The task of the odometry system is to keep track of the robot position using wheel encoder data. To implement this for a two wheeled robot the following formulas are used:

$$d_{left} = calculateMovementDelta(mtr0) \quad (5)$$

$$d_{right} = calculateMovementDelta(mtr1) \quad (6)$$

$$averageDistance = (d_{left} + d_{right})/2 \quad (7)$$

$$\delta\theta = atan2((d_{right} - d_{left}), wheelDis_{)}; \quad (8)$$

$$\delta x = averageDistance * cos(\theta); \quad (9)$$

$$\delta y = averageDistance * sin(\theta); \quad (10)$$

$$\theta + = \delta\theta \tag{11}$$
$$x + = \delta x \tag{12}$$
$$y + = \delta y \tag{13}$$

These are implemented in the x80 node.

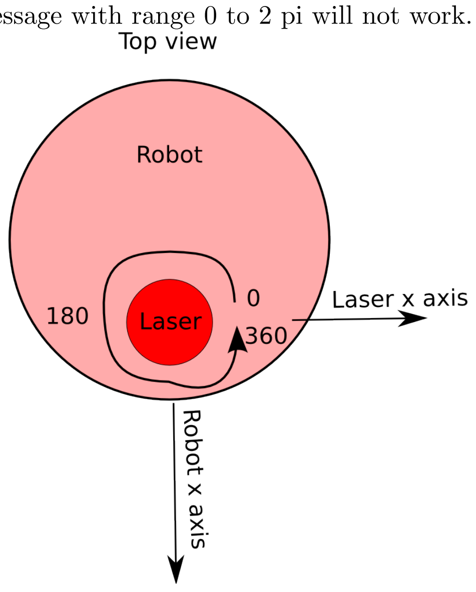
7 Slam

For slam the standard ROS node gmapping [1] can be used.

[1] http://wiki.ros.org/gmapping

7.1 Laser orientation

The orientation of the laser is important for the gmapping node. The node assumes that scan is symmetric around zero angle. This means that a laserscan message with range 0 to 2 pi will not work. A range from -pi to pi will work!



8 Navigation

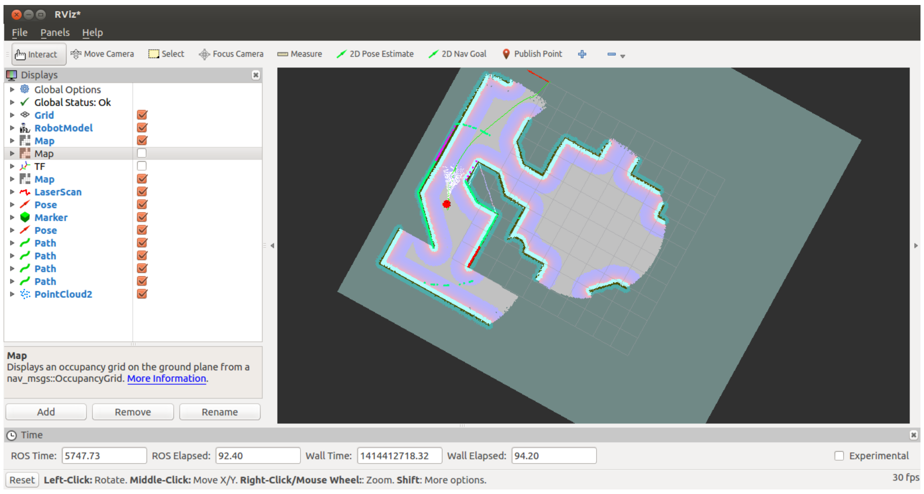
8.1 Framework

There is an existing framework for 2d mobile base navigation:

http://wiki.ros.org/move_base

Another framework is the skynav framework developed at saxion.

8.2 move base



Global Options
Global Status: Ok
Grid ✓
RobotModel ✓
Map ✓
Map ☐
TF ☐
Map ✓
LaserScan ✓
Pose ✓
Marker ✓
Pose ✓
Path ✓
Path ✓
Path ✓
Path ✓
PointCloud2 ✓

Map
Displays an occupancy grid on the ground plane from a nav_msgs::OccupancyGrid. More Information.

ROS Time: 5747.73 ROS Elapsed: 92.40 Wall Time: 1414412718.32 Wall Elapsed: 94.20 ☐ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 30 fps

9 Future work