Where Am I?

Saminda Abeyruwan

Abstract—Mobile robot localization estimates the robot pose relative to a given map of the environment. This project empirically evaluates the Adaptive Monte Carlo Localization (AMCL) algorithm using Gazebo simulator and RViz visualizer. Using ROS framework, a baseline two- and custom four-wheeled mobile robot platforms with senors have been developed. The robots have used ROS AMCL and navigation stack packages for global localization and path planning. By tuning the AMCL and planner parameters, the robots have been localized on the map and successfully navigated to predefined target locations.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, deep learning.

1 Introduction

MOBILE robot localization estimates the robot pose relative to a given map of the environment [1]. Given the coordinates of the map, a robot pose, $x_t = (x \; y \; \theta)^T$, sufficiently determines location and orientation of the robot. The robot pose cannot be directly sensed, and it must be inferred from the noisy sensor data over time. There are four dimensions with which the complexity of the localization problems can be analyzed.

The first dimension is based on the knowledge available initially, and during the operation time. There are three complexity classes; 1) *position tracking*, the initial pose of the robot is known, and the localization is achieved via accommodating the noisy robot motion. If the motion error is relatively small, the pose uncertainty is often approximated by a unimodal distribution. This is also a local localization, as the pose is confined to a small region of the robot true pose, 2) *global localization*, the initial pose of the robot is unknown, and it is inferred from relative to a known map of the environment. The problem is more difficult than position tracking, and generally cannot be bounded by a a unimodal distribution, and 3) *kidnapped robot problem*, the robot gets teleported to some other location. The robot needs to reevaluate its pose, and recover from such situations to exhibits truly autonomous capabilities.

The second dimension addresses the nature of the environment: 1) *static environments*, the objects in the environment remain static, and the only variable is the robot pose, and 2) *dynamic environments*, contains objects whose location of configuration changes overtime.

The third dimension addresses the ability of the localization algorithms to control the motion of the robot. There are two modes: 1) *passive localization*, where the localization module only observes the sensor and motion readings, and the robot is controlled by another set of modules, and 2) *active localization*, controls the robot to minimize the localization error.

Finally, fourth dimension involves the number of robots involved: 1) *single-robot localization*, all sensor and motion information is collected from a single robot platform, and 2) *multi-robot localization*, information from other robots in a team of robots are used to solve the localization problem.

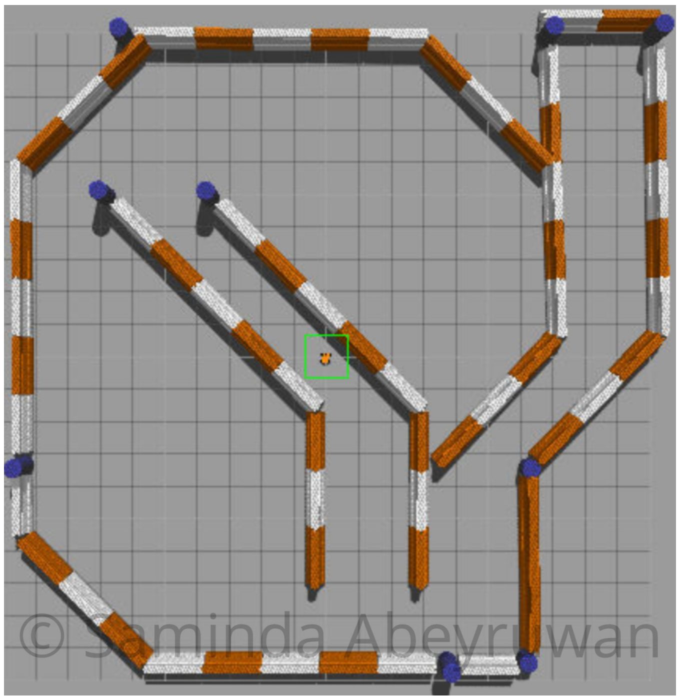


Fig. 1. Jackal race static map of the environment.

This project has focused on solving the global localization problem on a static environment, using a passive module, in a single-robot mobile platform. This has been implemented in simulation via ROS [2] packages to illustrate navigation tasks (Fig. 1).

2 Background

In global localization, the robot is placed within a known environment and localizes itself relative to an external reference frame. Due to the inherent uncertainty in sensor and motion measurements, the robot pose is represented by a *belief*, a probability density function over the all possible locations and orientation. Markov localization, an extension of the Bayes filter, provides a framework to obtain the robot pose as shown in the Alg. 1.

```
Algorithm 1 Markov_localization ($bel(x_{t-1}), u_t, z_t, m$)
  $\forall x_t$:
    $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}, m)\ bel(x_{t-1})\ dx_{t-1}$
    $bel(x_t) = \eta\ p(z_t|x_t, m)\ \overline{bel}(x_t)$
  return $bel(x_t)$
```

The belief of the robot at time t is represented by $bel(x_t)$. The measurement model and the motion model is given by $p(z_t|x_t, m)$, and $p(x_t|u_t, x_{t-1}, m)$ respectively. u_t provides the control command to move the robot from time t-1 to t, while z_t is the measurement, and m is the static map of the environment. Kalman filter and Monte Carlo localization are two specific instances of the realization of the Alg. 1, and further discussed herewith.

2.1 Kalman Filter Localization

The Kalman filter (KF) localization is a specific instance of Alg. 1, the belief, measurement, and motion models are represented by the first (mean) and second (covariance) moments. These assumptions limit the motion and measurement models to be linear, and the belief can only be represented by unimodal distribution such as a Gaussian.

The practical systems are inherently non-linear, and the two common filters used in practice are the Extended Kalman filter (EKF), and the Unscented Kalman filter (UKF) localization. The EKF uses Taylor seriese expansion to linearize the non-linear model equations, while UKF uses sigma points for the transformation. EKF and UKF localization represent with unimodal Gaussian and it is a good representation in uncertainty for tracking problems, but generally not suitable for global localization. While both algorithms addresses the limitation of the KF, the implementations use considerable computation resources (please refer to [1], chapter 7, for comprehensive description of the filters).

2.2 Monte Carlo Location

Monte Carlo Localization (MCL) represents the belief $bel(x_t)$ of the robot pose by particles, which is a popular algorithm used in global and kidnapped robot localization. MCL is computationally efficient than EKF and UKF, and it is not subjected to assumptions outlined in Sec. 2.1. The basic steps of the MCL algorithm are: 1) the initial global uncertainty is achieved via a set of pose particles sampled uniform randomly over the entire pose space, 2) the motion model moves the particles to the new locations, 3) the measurement model assigns important factors to each particle based on the observations, 4) the resampling setp samples particles relative to the importance weighs, and obtain the new belief, and set the new particles to uniform importance weights.

This project uses Adaptive Monte Carlo Localization (AMCL) algorithm, which is a variant of MCL, that uses Kullback-Leibler divergence (KDL)-sampling to adapt the size of the particle set over time.

2.3 Practical Considerations

Table 1 summarizes and compares the localization algorithms discussed in Sec. 2.1 and 2.2. While EKF and UKF are improvements over KF, they do come with limiting assumptions, and usage of considerable computational resources. On the other hand MCL and its variants can be used to represent multimodal distributions, and provide the ability to allocate proper computational resources.

TABLE 1
Comparison of EKF, UKF and MCL implementations

	EKF/UKF	MCL
Measurement	landmarks	row measurements
Measurement noise	Gaussian	any
Posterior	Gaussian	particles
Efficiency (memory)	++	+
Efficiency (time)	++	+
Ease of implementation	+	++
Resolution	++	+
Robustness	-	++
Global localization	no	yes

3 Robot Simulation

Robot Operating System (ROS) provides a set of tools and libraries to develop robot applications [2]. Using ROS constructs, two mobile robots have been developed. RViz has been use for visualization, and Gazebo has been used as the simulator.

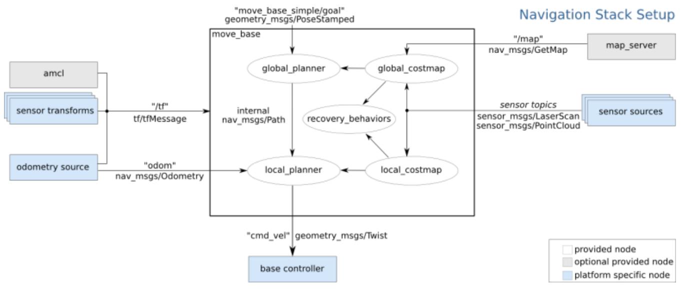


Fig. 2. AMCL and navigation stack setup.

AMCL[1] package and navigation stack [2] have been installed in a catkin workspace and setup according to Fig. 2. The simulation and parameter tuning were conducted in a Udacity virtual machine.

3.1 Udacity Bot

The baseline mobile robot, Udacity Bot (UB), is developed using Xacro[3] and Gazebo[4] definitions. The robot is constructed with two-wheels, a Hokuyo laser scanner[5], and a camera attached a chassis. The structure of the robot is show in Fig. 3, and it visual appearance is given in Fig. 4 . The chassis is a 0.4, 0.2, 0.1 box, while the wheels are cylinders with radius 0.1 and length 0.05. A differential drive has been attached to the left and right wheels, and configured via a Gazebo plugin.

1. http://wiki.ros.org/amcl
2. http://wiki.ros.org/navigation/Tutorials/RobotSetup
3. http://wiki.ros.org/xacro
4. http://gazebosim.org/tutorials/?tut=ros_urdf
5. http://gazebosim.org/tutorials?tut=ros_gzplugins

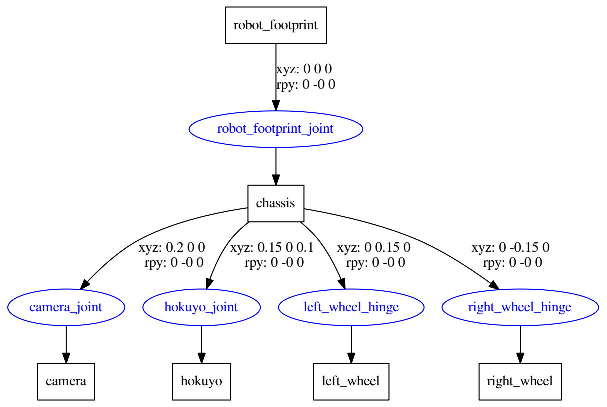


Fig. 3. Structure of Udacity Bot.

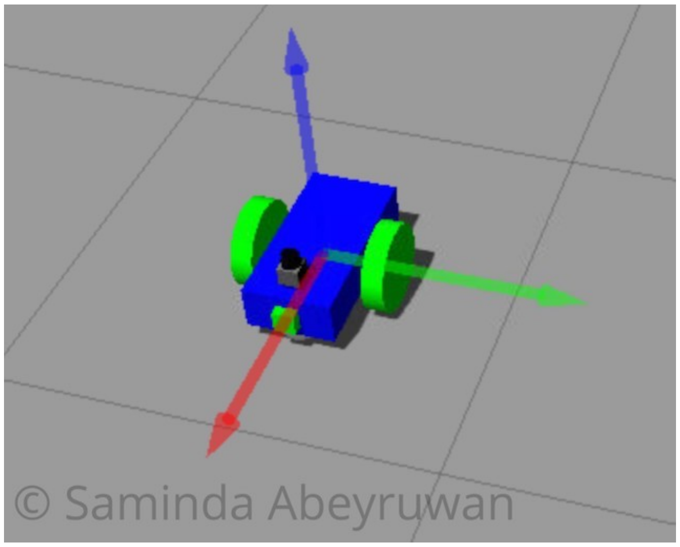


Fig. 4. Visual appearance of Udacity Bot.

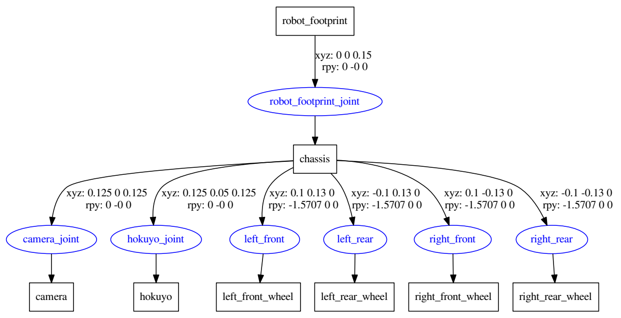


Fig. 5. Structure of Saminda Bot.

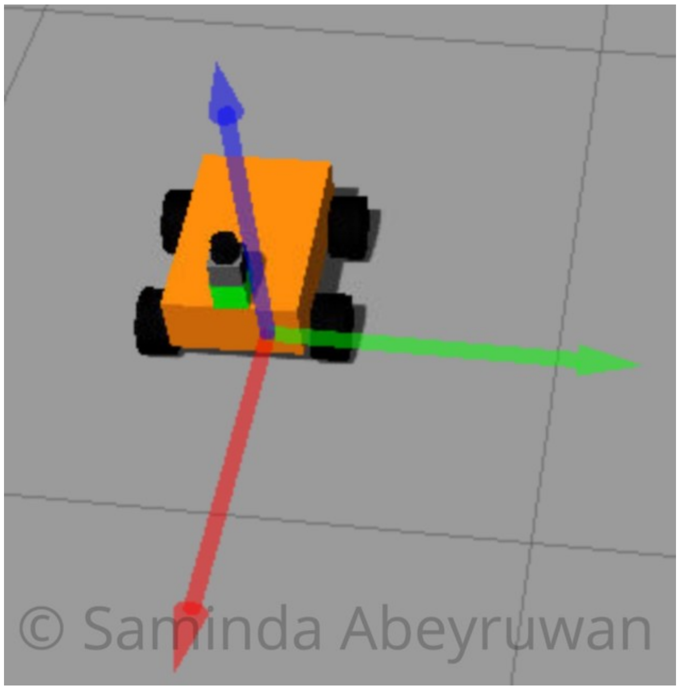


Fig. 6. Visual appearance of Saminda Bot.

3.2 Custom Robot Model

The custom mobile robot, named Saminda Bot (SB), is a four-wheeled robot. The chassis is a box 0.2, 0.3, 0.1, while the wheels are cylinders with radius 0.05 and length 0.05. Similar to Sec. 3.1, the robot is equipped with a Hokuyo laser scanner and a camera. The moment of inertia for each object shape has been calculated explicitly. A Skid Steering Drive has been attached to front/back left/right wheels and configured via Gazebo. Fig. 5 shows the structure of Saminda Bot, and the visual appearance is given in Fig. 6.

3.3 ROS Packages

The robots uses AMCL package and ROS navigation stack to localize, plan a path, and move to the target pose. In order to obtain successful robot behaviors, the parameters for the AMCL, global planner, local planner, global costmap, and local costmap have been changed (Fig. 2). Both robots have used the same set of parameters to configure AMCL (Table. 2). The navigation stack parameters have been configured to match the characteristics of each robot. This has been the most time consuming part of the project, and the ROS tool *rqt_reconfigure* has been used extensively to view all parameters and configure them in real-time without restarting the simulation. The parameters for local planner and costmap common is given in Table 3. The global and local costmap parameters have not been changed.

4 Results

Both the UB and SB robots have been localized and reached the benchmark target location. The two robots have different physical characteristics, and the navigation stack has been configured such that both robots complete the navigation tasks. The UB robot is more responsive and reached the target pose within an average of 75 seconds, while the SB robots has taken on average 158 seconds. Figure 9 shows

TABLE 2
AMCL parameters for both the UB and SB robots.

kld_err	0.05
kld_z	0.99
laser_lambda_short	0.1
laser_likelihood_max_dist	2.0
laser_max_beams	60
laser_model_type	likelihood_field
laser_sigma_hit	0.2
laser_z_hit	0.95
laser_z_short	0.1
laser_z_max	0.05
laser_z_rand	0.5
max_particles	200
min_particles	20
odom_alpha1	0.05
odom_alpha2	0.05
odom_alpha3	0.05
odom_alpha4	0.05
odom_alpha5	0.1
update_min_a	0.1
update_min_d	0.1
initial_pose_x	0
initial_pose_y	0
initial_pose_a	0

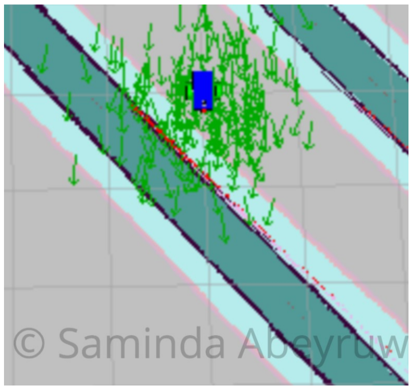
TABLE 3
Navigation stack parameters for the UB and SB robots.

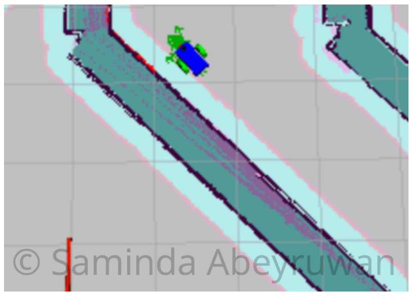
Local Planner Parameters	UB	SB
yaw_goal_tolerance	0.05	0.05
xy_goal_tolerance	0.1	0.1
pdist_scale	0.5	0.5
gdist_scale	1.0	1.0
max_vel_x	0.5	0.5
min_vel_x	0.1	0.1
max_vel_theta	2.0	0.45
min_vel_theta	-2.0	-0.45
acc_lim_theta	5.0	0.45
acc_lim_x	2.0	0.25
acc_lim_y	2.0	0.25
min_in_place_vel_theta	0.4	0.45
controller_frequency	10	10
Costmap Common Parameters		
obstacle_range	5.0	4.0
raytrace_range	8.0	6.0
transform_tolerance	0.2	0.2
robot_radius	0.25	-
footprint	-	[-0.2,-0.2],[-0.2,0.2], [0.2, 0.2], [0.2,-0.2]
inflation_radius	0.3	0.2

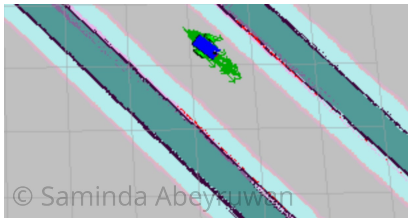
sample traces of the completion of the navigation tasks. The SB robot has been configured to move conservatively than the UB robot (Table 3). It has been observed that with the skid drive, if the robot turns faster, the AMCL injects particles into free space, and it has been controlled with slower motion. Figures 7 and 8 show the initial pose, three intermediate poses, and the final pose for the UB and SB mobile robots.

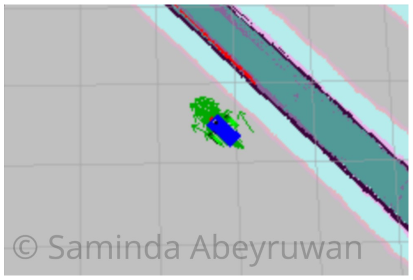
5 Discussion

Both robots have successfully localized and completed the navigation tasks. Due to the robots physical characteristics and different drives, the steering performance have empir-


(a) Initial pose


(b) Robot in motion (1)


(c) Robot in motion (2)


(d) Robot in motion (3)

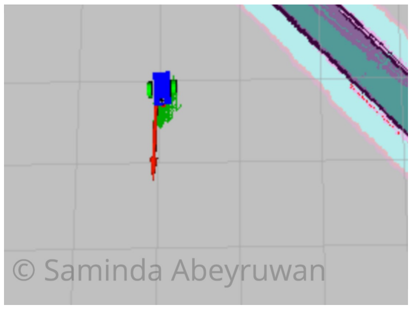
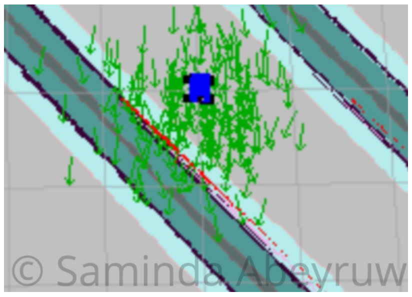
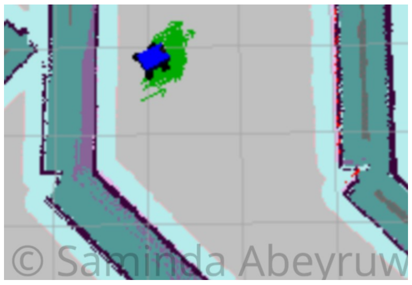

(e) Final pose

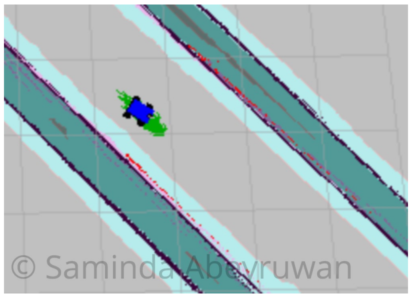
Fig. 7. Robot motion for the UB robot.



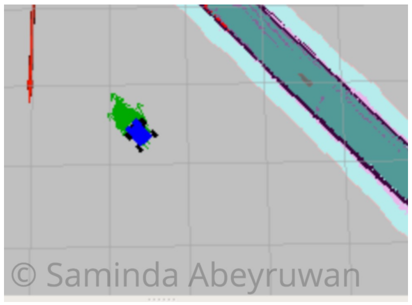
(a) Initial pose



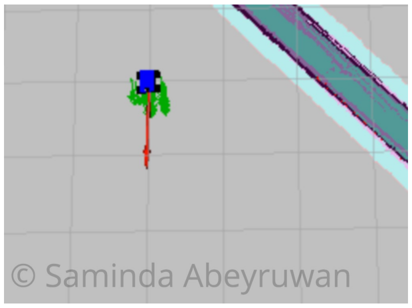
(b) Robot in motion (1)



(c) Robot in motion (2)

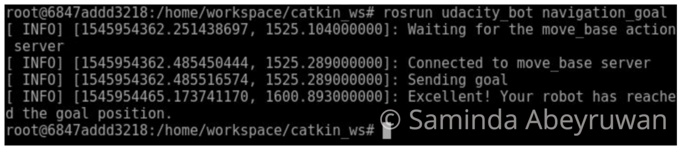


(d) Robot in motion (3)



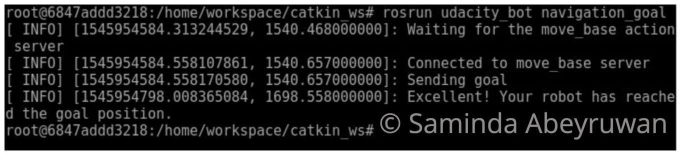
(e) Final pose

Fig. 8. Robot motion for the SB robot.



```
root@6847addd3218:/home/workspace/catkin_ws# rosrun udacity_bot navigation_goal
[ INFO] [1545954362.251438697, 1525.104000000]: Waiting for the move_base action
 server
[ INFO] [1545954362.485450444, 1525.289000000]: Connected to move_base server
[ INFO] [1545954362.485516574, 1525.289000000]: Sending goal
[ INFO] [1545954465.173741170, 1600.893000000]: Excellent! Your robot has reache
d the goal position.
root@6847addd3218:/home/workspace/catkin_ws# 
```

(a) A time trace for the UB navigation task



```
root@6847addd3218:/home/workspace/catkin_ws# rosrun udacity_bot navigation_goal
[ INFO] [1545954584.313244529, 1540.468000000]: Waiting for the move_base action
 server
[ INFO] [1545954584.558107861, 1540.657000000]: Connected to move_base server
[ INFO] [1545954584.558170580, 1540.657000000]: Sending goal
[ INFO] [1545954798.008365084, 1698.558000000]: Excellent! Your robot has reache
d the goal position.
root@6847addd3218:/home/workspace/catkin_ws#
```

(b) A time trace for the SB navigation task

Fig. 9. Time traces of navigation task.

ically shown to be significantly different. Due to the simulated noise in Gazebo, the robots have not taken the same path to reach the goal, and some paths were suboptimal and have taken more time to reach the target pose. AMCL has the capability to solve the kidnapped robot problem; it has been observed that when the robots were kidnapped, they were able to localize and move to the target pose.

6 Conclusion / Future work

The project evaluates the functions of the ROS navigation stack and AMCL packages on two simulated mobile robots. It has been shown that the robots were successfully localized and completed the navigation tasks. All tasks have been experimented in simulation. AMCL package and the navigation stack provide plethora of parameters that can be used to tune for the different robot platforms. It is also interesting to integrate the packages on real hardware and tune for the real world constrains. This project evaluated only one scenarios introduced in Sec. 1. It is advantageous to simulate other scenarios and use in real world robotic problems.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[2] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.