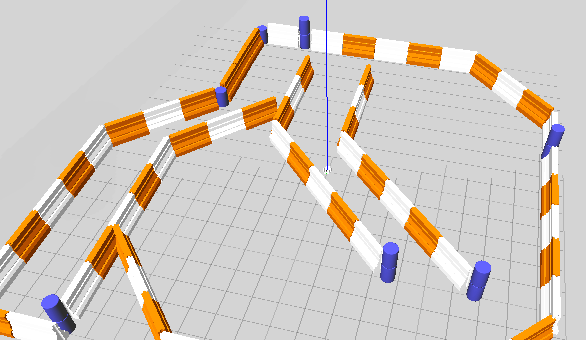
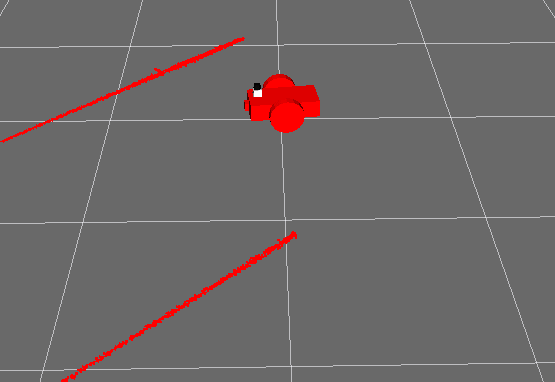
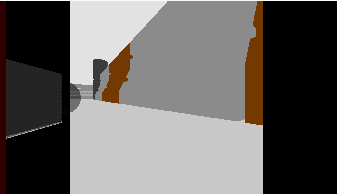
RoboND-Localization-Project

Report



Lutong Zhang

# 1. Abstract

# 2. Introduction

This report represents the Where AM I project, which simulate the localization problem of a small differential-wheels-driven robot with a camera and a lidar sensor on board in the simulation environment of Gazebo.

# 3. Background

The core of the Localization problem is to solve the question that where is the robot. Depending on the different situations, localization problems can be divided into three kind of localization problems: (1) local localization problem (position tracking), (2) global localization and the (3) kidnapped robot problem.

The goal of the local localization problem is to keep track of the position of the robot based on the information provided by different sorts of sensors. To get the accurate position of the robot, it needs to solve two major problems. The first one is the noise of the sensors, and the second one is the fusion of the data from different sensors. It is quite possible that the data from a noisy data yield the Gaussian Distribution. In the light of this statement, the noise of data can be measured by a mean and a standard deviation. The main goal of the global localization problem is to find the true location of the robot relative to the ground-truth map.

# 4. Model Configuration

## 4.1 Create a Robot Model

The robot model in the ROS system is represented by a Unified Robot Description Format (URDF), which is a XML formatted file [1]. A filed named as udacity\_bot.xacro was created under src/udacity\_bot/, which contains the discerption of the robot model. The robot is composed of a box body and two wheels, as well as two sensors on the box-shaped body as shown in the figure below.

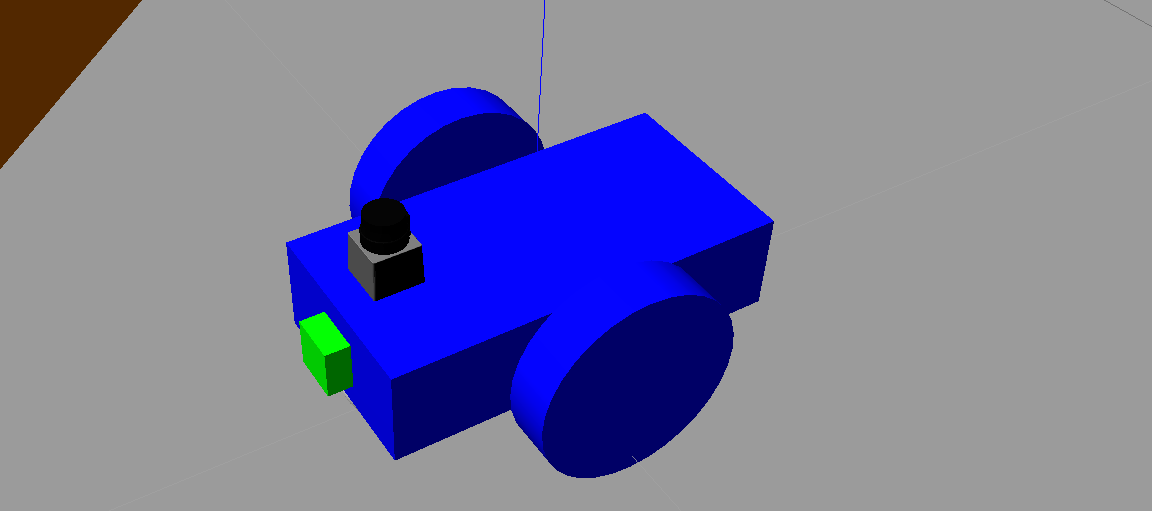


Fig. 1 Robot model

Basically, the robot model is a combination of several “links” connect by joints as described in the URDF file. In order to launch the robot model, a launch file was created and called by the udacity\_world.launch.

There are two sensors on the robot, a Hokuyo Lidar Rangefinder and a camera. The camera is simplified as a green box fixed to the front of the robot body, while the shape of the lidar is described by a mesh file (hokuyo.dae) which can be downloaded.

In order to collect sensor data such as images and laser-measured ranges, some plugins in Gazebo in employed. The plugin that drives the robot is called Differential Driver Controller, which is a controller that drives differential mobile base. It’s can be regarded as nothing but a ROS Node as well. The plug-in that support the camera and the lidar is pre-defined in Gazebo. The image taken by the camera is published to the image\_raw topic and the laser range data is published to the udacity\_bot/laser/scan topic. After all the setups and configurations, the simulation world and the robot model can be load by “*roslaunch udacity\_bot udacity\_world.launch*”.

## 4.2 Config the launch file for RViz

In the *robot\_description.launch* file, two nodes are added. The first one is the *joint\_state\_publisher* that publishes joint state messages for the robot. The second one is the *robot\_state\_publisher*, which handles the task of publishing the location and orientations of all the links in the robot to the transform tree. The transform tree is just another ROS node that keeps track of multiple coordinate frames over time and maintains the relationship between coordinate frames in a tree structure buffered in time [3], and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.

To start RViz together with Gazebo, “*<node name=”rviz” pkg=”rviz” respawn=”false”/>*” is added to the *udacity\_world.launch*. The robot and sensor data in Rviz are shown in the figure below.

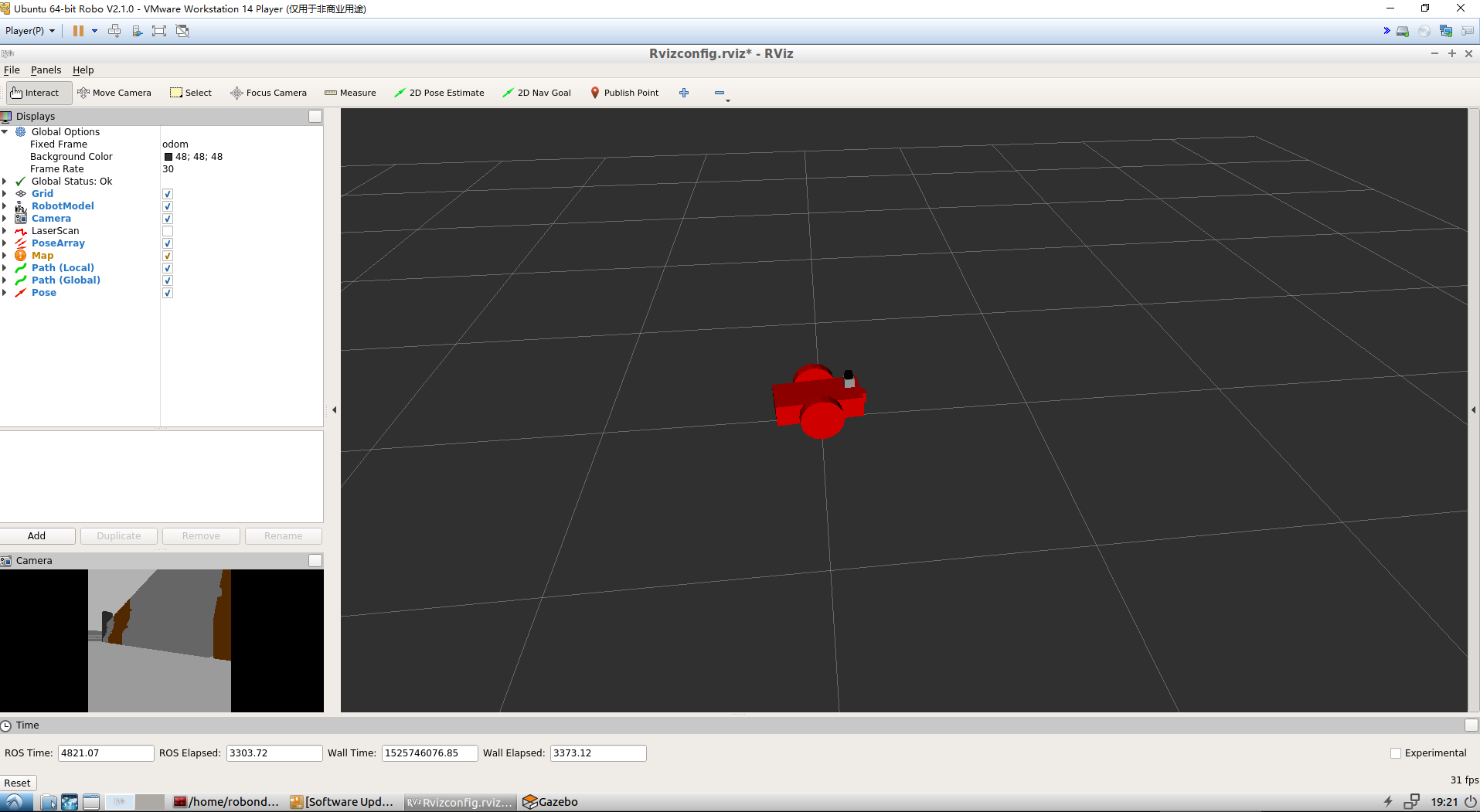


Fig. 2 Robot in RViz

## 4.3 Load the map

The map in the simulation environment is pre-constructed by Clearpath Robotics. This map is loaded by the *udacity\_world.launch* file.

## 4.4 Parameter Tuning

# 5. Results

# 6. Discussion

# 7. Future Work

Reference:

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

[2] http://wiki.ros.org/diff\_drive\_controller

[3] http://wiki.ros.org/tf