Udacity RoboND Project

# Deep Reinforcement-Learning Arm Manipulation

Final Project Report

Lutong Zhang

# 1. Abstract

# 2. Introduction

# 3. Background

## 3.1 A Brief Introduction

One fundamental desire for robotics automation is to let the robot make decisions and act based on the conditions of environment as well as its own condition. The behavior of the robot should be helpful or beneficial, in an optimized manner. In the context of reinforcement learning, this robot is named as an agent.

reward 
Agent 
observation 
action 
Environment 

Fig. 1 Interaction between Agent and Environment

The conditions of environment together with its own is named as state of the agent. The agent make action based on the state. Each action will lead to a result. The agent will be rewarded if the result is good, or other way around. A state of an agent is represented as , and its action can be represented as , where stands for index of time-point for discrete system. The reward of is usually written as .

The mechanism that the actions are chosen at different states is called a policy.

The return at time

In real world, the decision-making process of a robot is largely based on the pose of the robot and the map of the world around the robot. However, sometimes the map of the world around the robot and the pose of the robot relative to that map are not directly accessible, so the robot must build the map and find its pose based on the information of motion controls and measurements. In robotics, this problem is commonly abbreviated as SLAM (Simultaneous localization and map) problem, which is also known as concurrent mapping and localization(CML)[1].

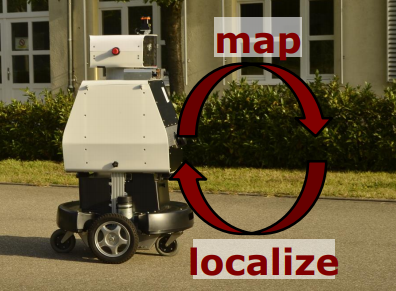


Fig. The chicken-or-egg problem of SLAM, figure from [3]

The standard from of the slam problem is stated as blow:

***Given:***

(a) Robot’s motion:

(1)

where means the motion of the robot from time to time .

(b) Robot’s measurement:

(2)

where means the measurement of the robot at time. The measurement is also named as observation of the robot.

***Wanted:***

(3)

(c) The map of the world:

(d) The path of the robot:

(4)

where is the pose of the robot.

## 3.2 Three forms of SLAM: EKF SLAM

The Extended Kalman Filter (EKF) SLAM is the first SLAM algorithm. It’s based on the EKF. The principle of the EKF is the Bayes filters. That is to say the measurements of sensors on the robot and the motions of the robot can be modeled by conditional Gaussian distributions.

In the EKF SLAM, the map of the world is feature-based. What the sensor observes are some raw data. Features can be extracted from these data. For each feature, its location, pose as well as it identity are saved in a vector. Some features are based on the range sensor data while some others may base on the RGB-D camera data, such as SIFT and SURF method. In robotics these features are called landmarks as well. Each feature extracted from the raw data has a unique signature, and the signature identify the features from on to another [1].

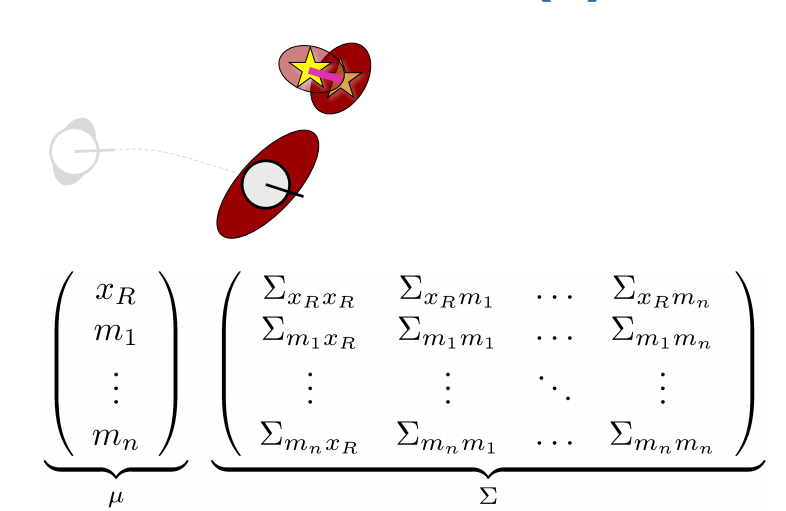


Fig.2 EKF Slam

In case of the feature-based map, the map is represented by a collection of landmarks. The landmarks are fixed on the map, while the features measured at different times may be different. At a certain time of , one to one correspondences between features and landmarks are denoted by the correspondence variable . just means that the ith feature is correspond to the jth landmark.

At time , with the robot pose described by , the control , the measurement and the correspondence variables , the estimation of the mean and c.o.v of location at time as well as the map, are given by the EKF SLAM algorithm with known correspondences [1].

## 3.3 Three forms of SLAM: Particle Filter SLAM

Fast-SLAM is based on particle filters. It’s a kind of non-parametric recursive Bayes filter. It works well in low-dimensional spaces and can model arbitrary distributions besides Gaussian distribution. It consists of three steps: sampling, importance weighting and resampling.

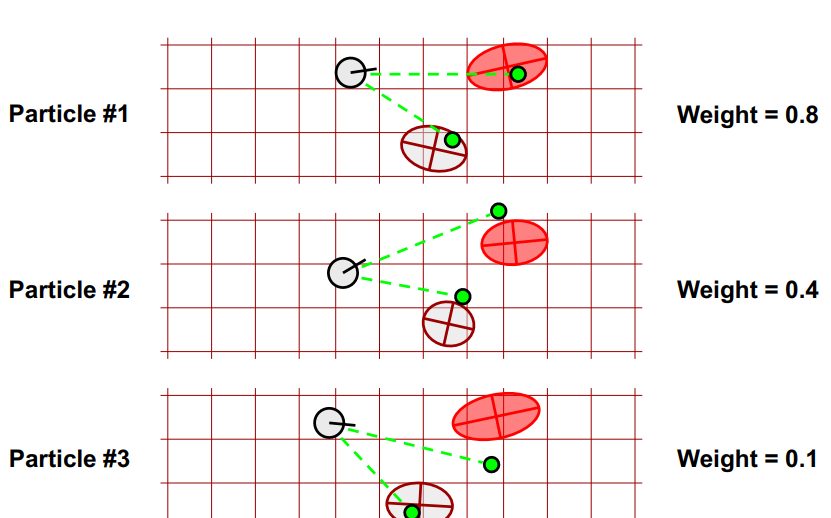
During the sampling step, the particles are sampled from the proposal distribution. The samples can be thought as one hypothesis about the state [3]. FastSLAM is a kind of particle filter SLAM algorithm. The first version of FastSLAM is proposed in 2002. In FastSLAM V1.0, each landmark is represented by a 2x2 EKF. For each particle, it needs to maintain the EKFs. The weight of each particle is updated by the error between the sensor measurement and the estimation of landmarks. Compared with v1.0, FastSLAM V2.0 also considers the measurements during sampling. 

Fig. 3 Particle SLAM

## 3.4 Three forms of SLAM: Graph-based SLAM

The graph-based SLAM is another way to solve the SLAM problem. A graph is a computational model that is composed of nodes connected with edges. The edges here are called constraints, which are constrained by the physical world. As its name, the graph-based SLAM is based on a graph.

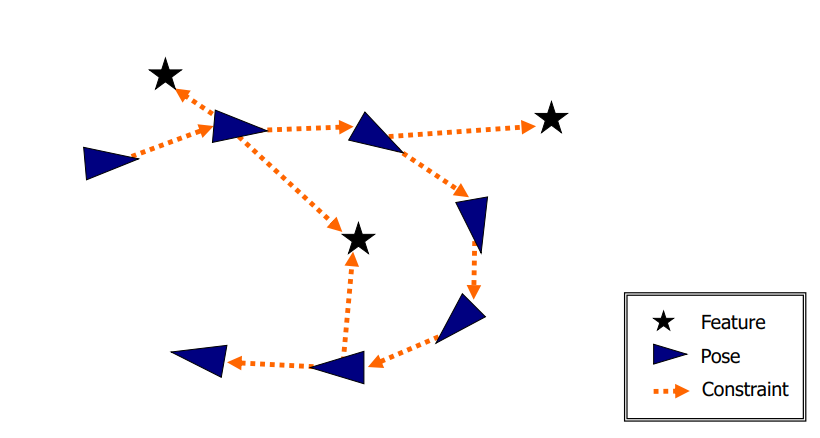


Fig. 4 Graph composed of nodes and constraints [2]

Each node in the graph represents a robot position and a measurement acquired at the position. The edge between two nodes represents a spatial constraint relating the two robot poses. A constraint consists in a probability distribution over the relative transformations between the two poses. These transformations are either odometry measurements between sequential robot positions or are determined by aligning the observations acquired at the two robot locations [2], as in Fig. 4.

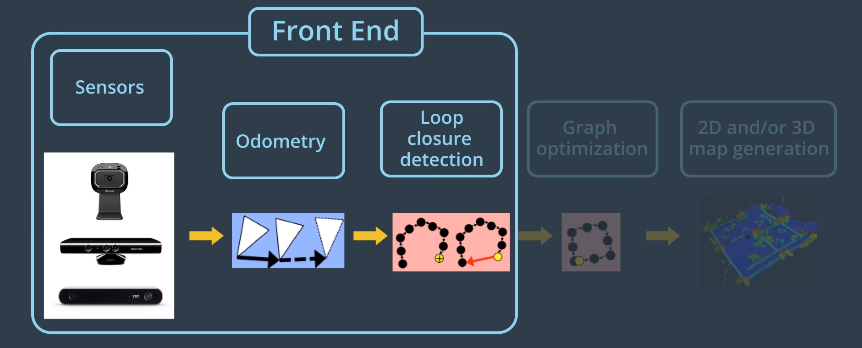


Fig. 5 Front-end in Graph SLAM [4]

The process mentioned above is called the front-end of the Graph-based SLAM. The main jobs of the front-end is to process raw data collected from all the sensors, to abstract features and to form the constraints between the nodes, as shown in Fig. 5. There are two kinds of constraints: the odometry constraint and the loop closure constraints. To form a loop-closure constraint, the front-end must have the ability to detect loop-closure as well.

The main jobs of the back-end of Graph-base SLAM is to determine the most likely configurations of the poses given the edges of the graph, as shown in Fig. 6. Another job for the back-end is to reconstruct the robot poses and generate the 2D/3D map.

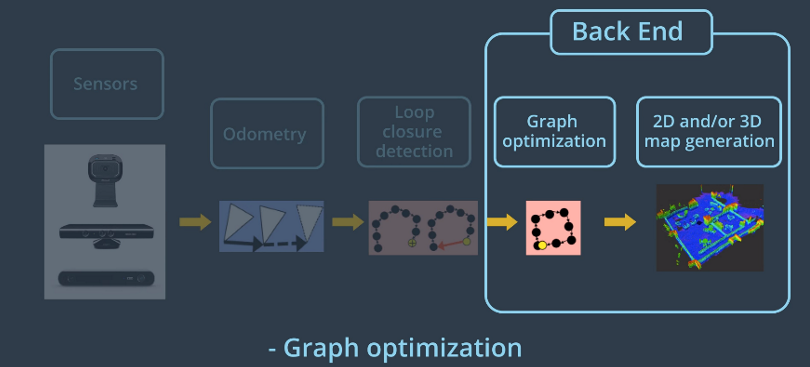


Fig. 6 Back-end in Graph SLAM [4]

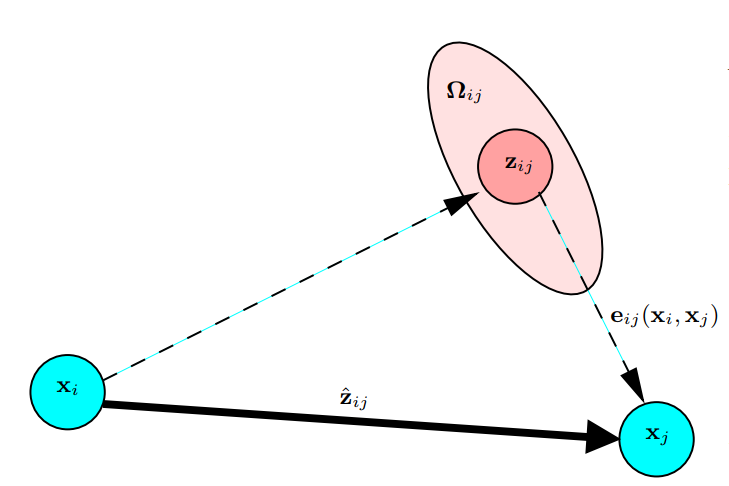


Fig. 7 Constraints among nodes [2]

Let describe the pose of node . The Vector of parameters should be

(5)

The mean and the information matrix of the virtual measurement between node and node are represented by and . Suppose the prediction of the transformation from to is given by . The log-likelihood of a measurement thus is

(6)

Assume

(7)

Thus the total likelihood function should be

(8)

where is the set of node pairs between which the exists. The final goal of the whole setup is to find the configuration of the nodes that minimize the negative log likelihood . That is to say,

(9)

One of the major jobs of back-end is to find the **.** After this process, the best estimation of the map can be generated based on **.**

## 3.5 RTAB Map

The RTAB-Map is an implementation of the Graph-based SLAM. Beside the very basic ideas of the Graph-based SLAM, RTAB-MAP employs loop-closure as constraint as well, which well be explained in detail below.

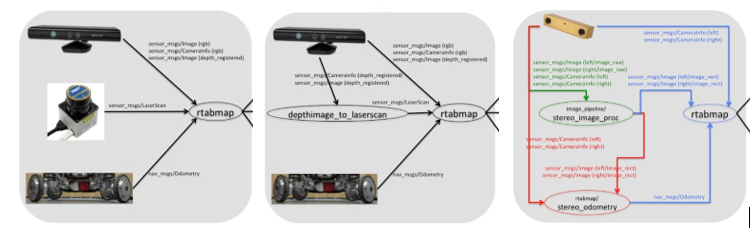


Fig. 8 Different setups for RTAB-MAP [5]

As shown in Fig. 8, there are several different setups for the RTAB map. The basic setup utilizes a RGB-D camera, a laser scanner and Odometry data. The RTAB-MAP largely depends on the image data for feature abstraction and loop-closure detection. When the robot moves, a feature extraction algorithm called SURF is processing the RGB image collected by the RGB-D camera. For each location, it is combined with an image signature, a time index and a weight. As shown in the Fig. 9, the nodes are connected by odometry constraints, measurement constraints as well as the loop closure constraints.

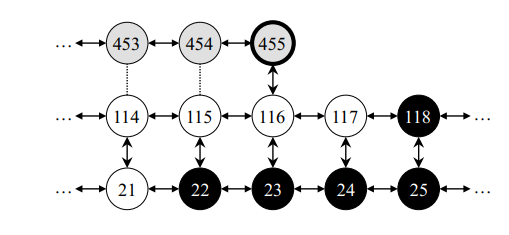


Fig. 9 Nodes(Locations) in the RTAB-MAP [7]

Thus the process of RTAB-MAP can be explained as the following.

### 3.5.1 Raw Image Processing and Feature Extraction

The RGB-D data is collected by the camera. The RGB image frame processed by the SURF method. The SURF uses wavelet responses in both horizontal and vertical directions, then the Gaussian filter is used. At the end, the SURF extracts features from the image and gives each feature a descriptor. Using the bag-of-words approach, a unique signature is associated with the image. The signature is represented by a set of visual words contained in a visual vocabulary incrementally constructed as time flies [7]. Each visual word of the vocabulary refers to a single feature’s descriptor (a vector of 64 dimensions). To compare the descriptors of the features, a randomized forest of four kd-trees is used [7]. This approach increases the efficiency of nearest-neighbor search because the trees are build from all the SURF feature descriptors of the words contained in the vocabulary. When the criterion is not satisfied, a new word is created with the feature’s descriptor. The new word is then added to the vocabulary and the signature of the current image.

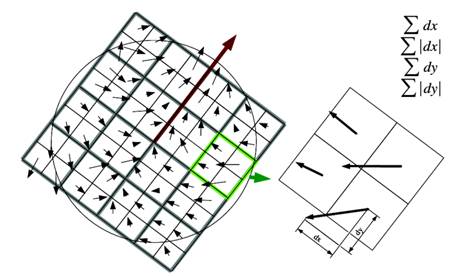


Fig. 10 SURF feature extraction [8]

Once a new location is set up, its signature is checked firstly. If the quality of its signature is bad, the data is then dropped. If its signature passes the checking process, the new location is created and been associated with a signature and a time index. Finally, this newly created location is passed into the Short-Term Memory(STM).

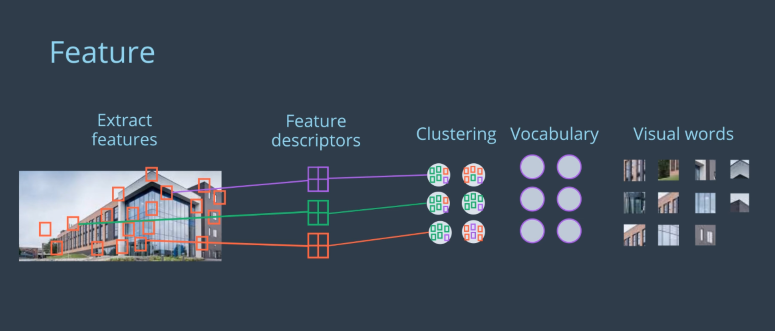


Fig. 11 Bag-of-words approach [4]

### 3.5.2 Weight Update

When the newly created location () is piped into the STM, it is compared with the last location in the SMT (). By comparing it means that the ratio of the number the similar words and the number of the total words (the bigger on of the two locations) is calculate. When the ratio is too high, the is merged into , which means that only the from is kept and the newly created words with is removed from the vocabulary. In a simple way, the signature of is cleared and signature of is copied to the signature of . After the merging process, the weight of is increased by the weight of plus one, and the constraints in the graph of are redirected to . After that, the is deleted from STM.

### 3.5.3 Bayesian Filter Update

Loop closure hypotheses is check by the Bayesian filter to see if matches one of the visited locations stored in WM (working memory).

When a loop closure hypothesis is accepted, is linked with the old location . The weight of is increased by add the weight of . Then the weight of is reset to 0, and a loop closure link is added between and .

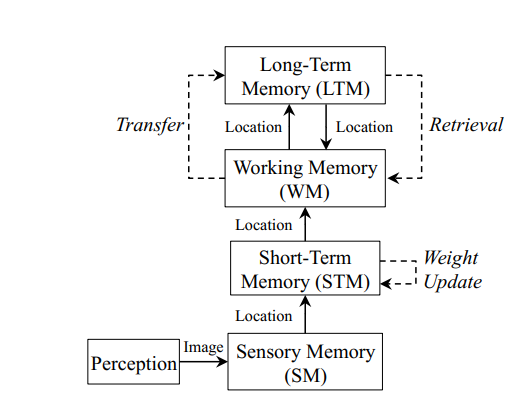


Fig. 12 Memory Management of RTAB-MAP

### 3.5.4 Retrieval

If the loop closure is established, the location with the highest loop closure hypothesis are transferred back from LTM (Long Time memory) back to WM. Meanwhile, the associated visual words are also retrieved. The references are added which links the retrieved visual words and the signatures. For the words that are not in the bag, their descriptors are quantized to the random forest. As this process is computational time consuming, on two locations are retrieved from the LTM.

### 3.5.5 Transfer

When processing time for an image is greater than , the oldest locations of the lowest weighted ones are transferred from WM to LTM [7]. The configuration of is depend on the CPU capability of the robot.

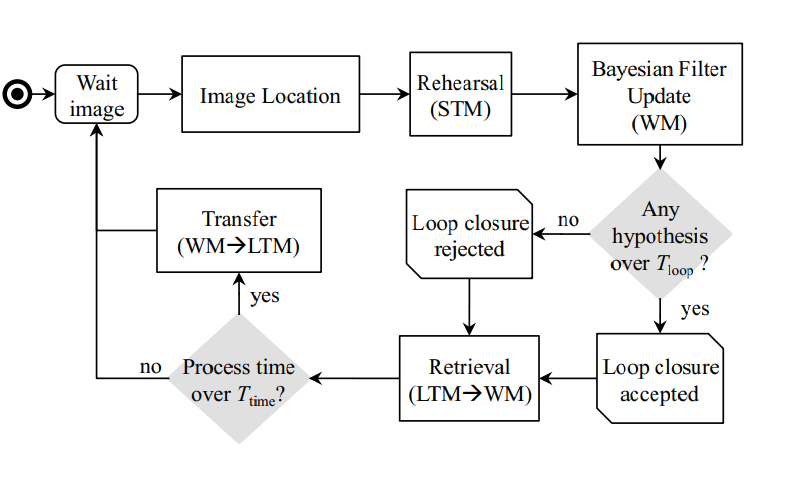


Fig. 13 The RTAB-MAP algorithm

### 3.5.5 Graph Optimization

The RTAB-MAP provides three kind of optimization method. The first one is named as Tree-based network optimizer (TORO). The second one is the General Graph Optimization method. The last one is the G2O or the GTSAM method.

After the graph optimization is done, the RTAB-MAP can output the path of the robot, the 2D map and the 3D map as shown in Fig. 14.

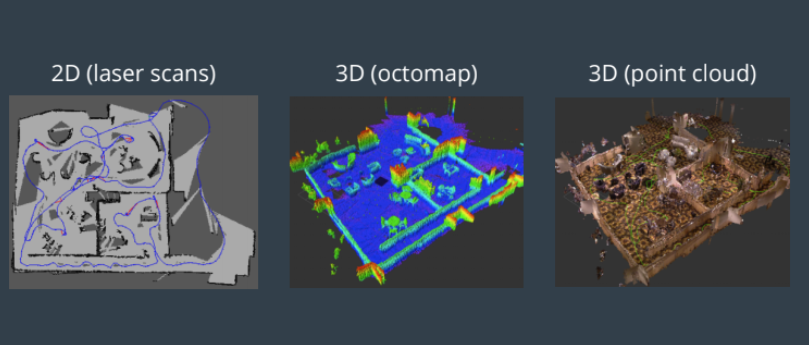


Fig. 14 Output of RTAB-MAP

# 4. Scene and Robot Configuration

This chapter mainly introduces the details of the setup of the project.

## 4.1 Setup of the Robot

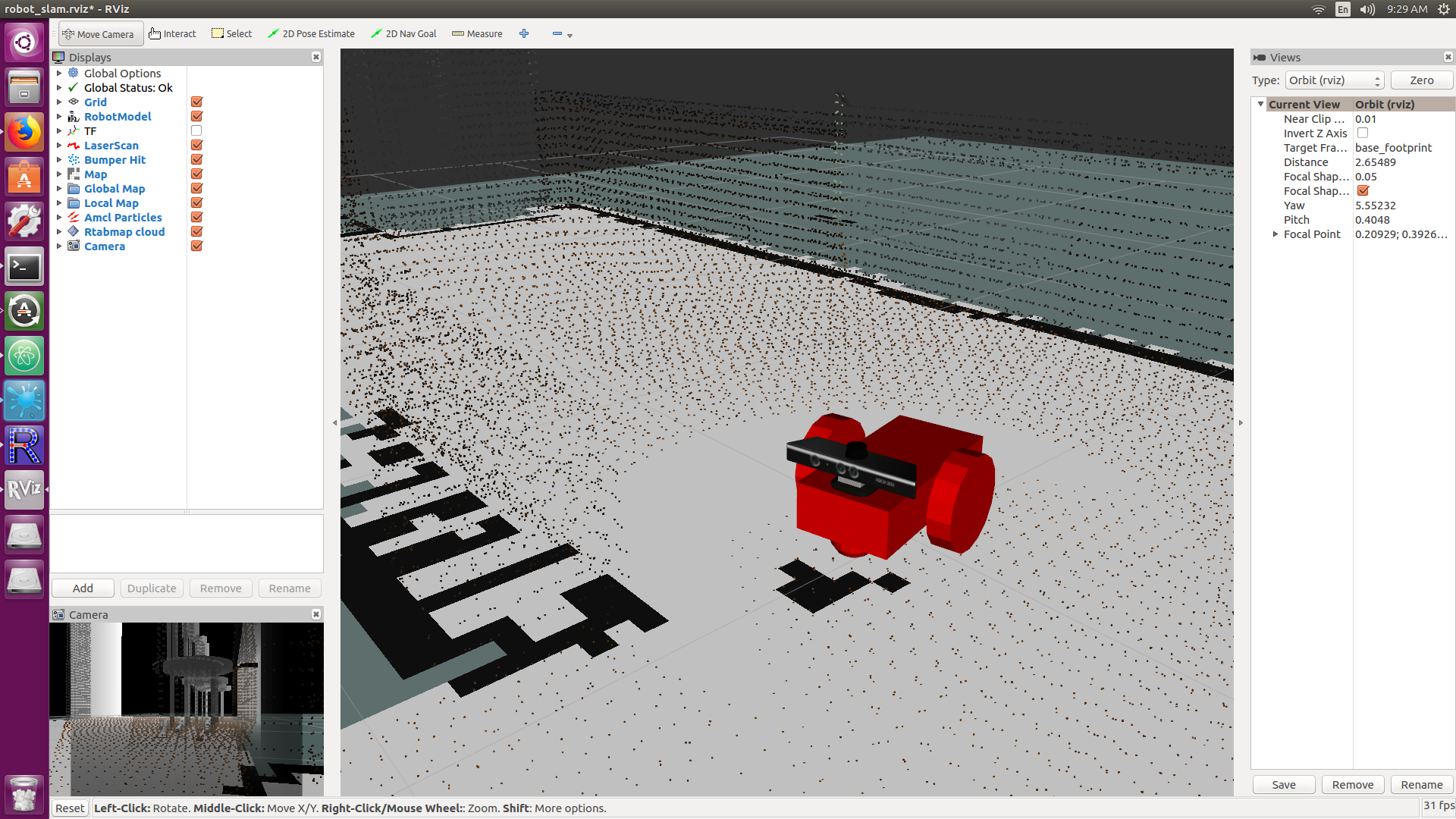


Fig. 15 The robot

The robot for the Gazebo simulation environment is build up based on the robot description files under */urdf*. This robot is modified from the robot used in the previous project.

There are two files in the folder. */urdf/udacity\_bot.xacro* builds up all the components of the robot such as the body, the sensors and the wheel. They are defined as links in the file, and they are connected by joints. The visual model of the sensors are imported from the mesh files download from the internet.

*/urdf /udacity\_bot.gazebe* imports the Gazebo plug-ins for differential-dirvers, RGB-D camera and the Hokuyo laser scanner. It defines the color of the robot as well. When importing the plug-ins, this file also defines their parameters.

## 4.2 Launch Files

Under */launch*, there are totally five launch files.

* */launch/world.launch* is the file which load the world to the simulator and spawn the robot in the simulator.
* */launch/robot\_description.launch* is loaded by the *world.launch* file. The function of this file is to tell the simulator to spawn the robot from .xacro file, load the *joint\_state\_publisher* node and the *robot\_state\_publisher* node.
* */launch/teleop.launch* launches the node of *teleop*, which receives the odometry control command from keyboard and send it to the robot.
* */launch/rivz.launch* launches the rivz and do the configuration of the rivz.
* */launch/mapping.launch* is the file which launch the *rtabmap* node and the *rtabmapviz* node. The parameters of the rtabmap are configured within this file. The remap keyword maps the original-name to the new-name. Each time a node uses any of its remapping’s original names, the ROS client library silently replaces it with the new name from that remapping.

In the rtabmap node, the names of the sensor data topics are remapped to get sensor message from the Gazebo node. After the remapping, some of the parameters in the rtabmap node are modified to better fit this project.

The remapping is also implemented for the rtabmap\_ros node, so that the sensor data can be feed into this node.

## 4.3 Overall Structure of the Project

# 5. Results

# 6. Discussion

# 7. Future Work

# Reference:

**[1]** T. Sebastian, B. Wolfra, F. Dieter, PROBABILISTIC ROBOTICS, 1999-2000

**[2]** G. Giorgio, K. Rainer, S. Cyrill Stachniss, B. Wolfram, A Tutorial on Graph-Based SLAM, Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany

**[3]**http://ais.informatik.uni-freiburg.de/teaching/ws17/mapping/pdf/slam05-ekf-slam.pdf

**[4]** Udacity-RoboND, Lecture 18 GraphSLAM

**[5]**http://wiki.ros.org/rtabmap\_ros/Tutorials/SetupOnYourRobot#Bring-up\_your\_robot

**[6]**https://introlab.3it.usherbrooke.ca/mediawiki-introlab/images/3/31/Labbe2015ULaval.pdf

**[7]** L. Mathieu, Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation, IEEE, Franc¸ois Michaud, Member, IEEE

**[8]**https://courses.cs.washington.edu/courses/cse576/13sp/projects/project1/artifacts/woodrc/index.htm