Manual of Kidnapped robot problem localization system

Thai An Le

September 7, 2019

1 Implementation Details

1.1 Mathematical Backgrounds

This section proposes the mathematical base for Wifi Received Signal Strength Indication (RSSI) sensor model in Kidnapped robot problem (KRP) localization system. The sensor model makes no assumptions about the number of Wifi Access Points (APs) as well astheir coordinates in the map. Although the distribution of APs may affect the performance of the learning system, the model generally does not assume AP distribution. However, the model does assume APs having 802.11 capability and the direction of RSSI measurement does not affect its result signal strength.

1.1.1 Path Loss

The main idea is to provide a mapping function between Wifi signal strength prediction and robot position hypotheses as particles. We characterize the mapping function by a simplified path loss model and consider shadowing and multipath as system noise. These simplified path loss models only aim to capture the essence of signal propagation without resorting to complex ray-tracing models or other geometric considerations. We define that Path Loss model(Miyagusuku, Yamashita and Asama, 2018) as the follow equation with each location x and each individual AP j:

$$\begin{aligned} PL_j(x) &= P_{0j} - k_j log_{10}(d_j) + \epsilon \\ d_j &= \left\| x - (apx_j, apy_j) \right\| \end{aligned} \tag{1}$$

where P_{0j} is the RSSI measurement 1 meter from AP j, (apx_j, apy_j) is the AP x-y coordinate in static map, k_j is the path loss exponent and ϵ is Gaussian noise with known variance σ^2_{PL}. As mentioned, we do not make assumptions about the AP coordinates as well as propagation environment. Hence, the parameter vector $\theta_{PL,j} = (P_{0j}, apx_j, apy_j, k_j)$ for each AP j is learned by fitting

the negative log likelihood $nll_{PL_j}(x)$ of training data given Path Loss model parameters:

$$nll_{PL_j}(x) = 0.5nlog(2\pi) + \sum_{i=1}^{n} log\sigma_{i,j} + \sum_{i=1}^{n} \frac{(z_{i,j} - PL_j(x_j, \theta_{PL_j}))^2}{2\sigma_{i,j}^2} \tag{2}$$

for n training data points, where $\sigma_{i,j}^2 = \sigma_{PL,j}^2$ and in case of zero value training output, the variance is penalized to be higher $\sigma_{i,j}^2 = \sigma_{PL,j}^2 + \sigma_{zero}^2$. The negative log likelihood is then optimized by gradient descent.

1.1.2 Noise Model

To learn the shadowing and multipath noise components in Wifi signal strength, we characterize these noise components as the different between measurement z and the Path Loss model $PL(x)$:

$$ze = z - PL(x) \tag{3}$$

Then, a GP model is used to learn the Equation 3 for all APs as specified in (Le, 2019). This GP model will output predicted noise mean $E[ze_{*,j}]$ and predicted noise variance $var(ze_{*,j})$ according to Equation 4. The stared variable implies new input other than training data.

$$\begin{aligned} E[ze_*] &= k_*^T(K + \sigma_n^2 I_n)^{-1} z \\ var(ze_*) &= k_{**} - k_*^T(K + \sigma_n^2 I_n)^{-1} k_* \end{aligned} \tag{4}$$

with $K = cov(X, X)$ being a $n \times n$ covariance matrix between all training points $x_i \in X$, $k = cov(X, x_*)$ the covariance vector of all training points to x_*, and $k_{**} = cov(x_*, x_*)$ the variance of test point x_*.

1.1.3 Wifi RSSI Sensor Model

Finally, the predicted mean signal strength $E[z_{*,j}]$ for each AP j given test point x_* is defined as the sum of Path Loss model $PL_j(x_*)$ and predicted noise mean $E[ze_{*,j}]$, bounded by zero:

$$E[z_{*,j}] = max(PL_j(x_*) + E[ze_{*,j}], 0) \tag{5}$$

and the variance of sensor likelihood is the same with sensor noise variance $var(ze_{*,j})$:

$$var(z_{*,j}) = var(ze_{*,j}) \tag{6}$$

Given these rectified statistics, the likelihood for each individual AP j can be computed. We therefore state the sensor model likelihood for each AP j as standard distribution:

$$p(z_j|x_*) = \Phi(\frac{z_j - E[z_{*,j}]}{var(z_{*,j})}) \tag{7}$$

To use our model as the perceptual likelihood, we finally combine likelihood of each individual AP sensor model by geometric average, given the test location x_*:

$$p(z|x_*) = (\prod_{j=1}^{m} p(z_j|x_*))^{\frac{1}{m}} \tag{8}$$

The Equation 8 assume that m individual APs likelihood are statistically independent. The final sensor model likelihood is suitable to plug in MCL algorithm for tracking localization or provide an initial prior distribution for faster global localization.

1.2 Process Flows

This section assumes that the model has been trained to learn the parameters $\theta_{PL,j} = (P_{0j}, apx_j, apy_j, k_j)$ for each AP j and the GP model parameters, and these parameters are stored. From the main Equation 8 for Wifi Sensor likelihood, we could:

- **Generate Wifi Particle cloud as prior distribution for faster global localization on amcl**. This process is depicted in Figure 1.
- **Detect KRP incident and hence re-localize robot pose in case of fail localization**. This process is depicted in Figure 2.

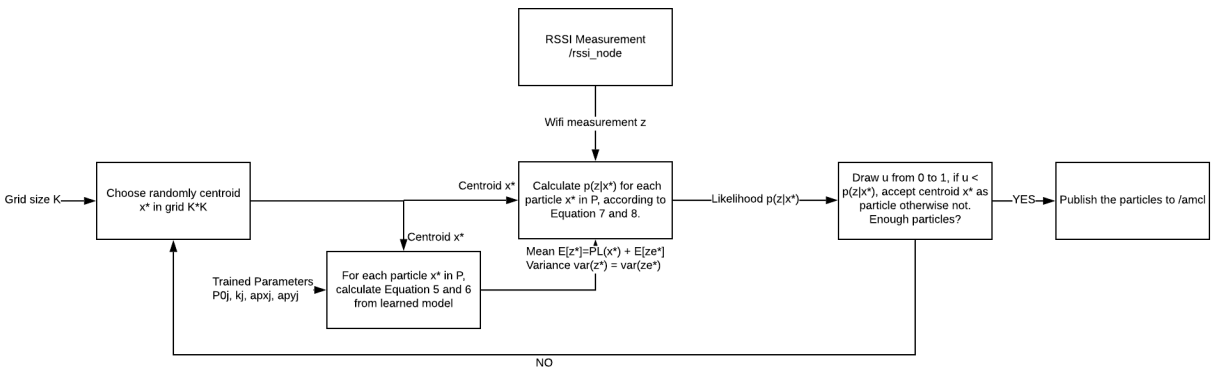


Figure 1: Process flow (1) diagram of Wifi Particles Generation

In process flow 1, to generate wifi particles P_{new} for **amcl** localization system, we first discretize the map into a grid $K * K$ with user-defined grid size K. Then, we randomly choose a cell with centroid x_* for predicting mean $E[z_{*,j}]$ and variance $var(z_{*,j})$ wifi signal strength at that particular centroid x_*, using trained Path loss and noise model. At the same time, we measure the real wifi signal strength array z, which consists of independent AP signal strength z_j. Subsequently, according to Equation 7, we use wifi signal strength array z and the computed mean $E[z_{*,j}]$ and variance $var(z_{*,j})$ to estimate final sensor likelihood $p(z|x_*)$ for centroid x_*. Next, we use the likelihood $p(z|x_*)$ to assess centroid x_* to be accepted as particle or not. To do this, an random variable u is drawn in range from 0 to 1 and then compared with the likelihood $p(z|x_*)$, if $u < p(z|x_*)$ then centroid x_* is accepted as particle and otherwise not. The particle generation is continued until enough particles are produced. The final particle set P_{new} is concentrated around true robot pose and hence faster global localization is achieved.

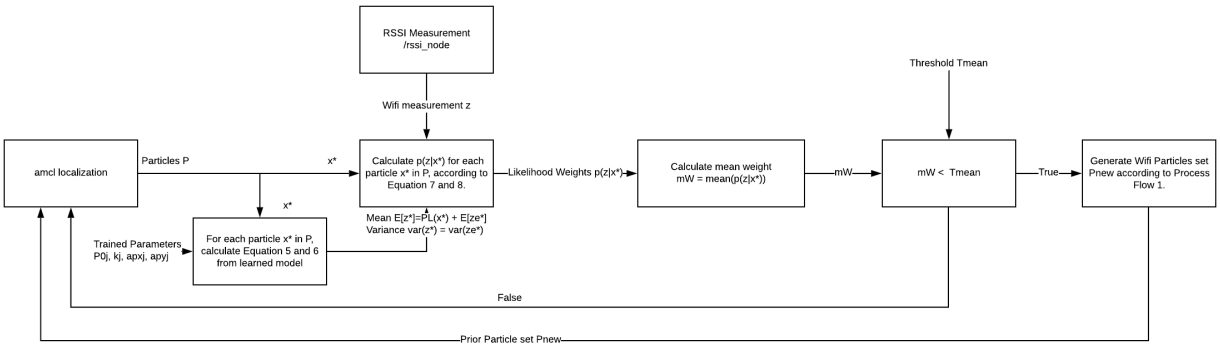


Figure 2: Process flow (2) diagram of KRP detection algorithm

In process flow 2, using wifi sensor likelihood $p(z|x_*)$, we assess the laser particles P to judge if P is good enough to be consider accurate robot distribution. First, we queries the laser particles P from **amcl**. For each particle x_*, we also predict mean $E[z_{*,j}]$ and variance $var(z_{*,j})$ wifi signal strength using trained model. We also measure the real wifi signal strength array z at that time. After that, for all particle x_* in laser particle P, we compute likelihood weight array $p(z|x_*)$ and then compute the mean particle weight:

$$mW = \frac{1}{|P|} \sum_{x_* \in P} p(z|x_*) \tag{9}$$

Finally, if mW is less than a user-defined threshold $Tmean$, which means the KRP incident has happened, we trigger the wifi particle generation as defined in Process flow 1 and input the new particle set P_{new} back to **amcl**. This will assist the re-localization process to be faster and the new

robot belief distribution is again accurate. On the other hand, if $mW \geq Tmean$, the system judges that laser particles is still accurate to represent true robot belief distribution and the navigation continues.

2 Installation and Configurations

2.1 Installation

This section assumes the users have followed the ROS Kinetic installation instructions in this link: http://wiki.ros.org/kinetic/Installation/Ubuntu, to install on a workstation that will connect to **youbot** mobile robot with the same network. The current state of **youbot** mobile robot is configured to work with my **krp_localization** package, any further configurations requires understanding of my project and expertise.

2.1.1 Resolve Dependencies

Currently, **krp_localization** package stack is implemented in Python to harness its rich mathematical libraries such as numpy, scipy to easily work with matrices and probabilistic models. Open a terminal and install these required libraries:

```
pip install numpy==1.12, decorator
pip install scipy==0.18
pip install --no-deps GPy, paramz
```

Some ROS packages may be required:

```
sudo apt install ros-kinetic-tf2-sensor-msgs, ros-kinetic-robot-pose-ekf
```

2.1.2 Create workspace and download packages

Because of considerable amount of packages in my **krp_localization** package stack, it is recommended to create a new ROS workspace for isolating other packages. Now we will create new directory for the workspace:

```
mkdir -p ~/robotics_ws/src
cd ~/robotics_ws
```

and initialize the workspace:

```
catkin init
catkin config --extend /opt/ros/kinetic
git config --global credential.helper 'cache --timeout=120'
```

```
wstool init

echo "source ~/robotics_ws/devel/setup.bash" >> ~/.bashrc
```

We will download the relevant packages for **krp_localization** package stack into the workspace. Firstly, navigation package is required from my repository.

```
cd ~/robotics_ws/src
git clone https://github.com/anindex/navigation
cd navigation
git checkout kinetic-devel
```

then we will download the **youbot_description**, **youbot_navigation** package for both simulation and real world controller package for **youbot**. we also download the main package for KRP localization system named **krp_localization**.

```
cd ~/robotics_ws/src
git clone https://github.com/anindex/youbot_description
git clone https://github.com/anindex/youbot_navigation
git clone https://github.com/anindex/krp_localization
```

2.2 Compiling and Configuration

For using, we will compile and source the workspace in which we downloaded previous packages:

```
cd ~/robotics_ws
catkin build

source ./devel/setup.bash
```

and setup running permissions for each Python ROS nodes:

```
cd ~/robotics_ws/src/
chmod +x krp_localization/src/rssi_node
chmod +x krp_localization/src/rssi_localization_server
chmod +x krp_localization/src/prob_mesh_visualizer
chmod +x krp_localization/src/record_data
chmod +x krp_localization/src/krp_localization_rssi
```

In order to communicate with **youbot**, users also need to configure environment variables such as ROS_IP and ROS_MASTER_URI corresponding to its static IP configuration. We will now setup these variables:

```
echo "10.12.0.2 youbot" >> /etc/hosts

echo "export ROS_IP=<enter your IP here>" >> ~/.bashrc
```

```
echo "export ROS_MASTER_URI=http://youbot:11311" >> ~/.bashrc
```

On the laptop that used to measure Wifi RSSI measurement, users must give root privilege for **iw, iwlist** tool, otherwise they will output wrong results.

```
sudo chmod 4777 /sbin/iw
sudo chmod 4777 /sbin/iwlist
```

3 Collecting Training data

This appendix serves as tutorial for users how to collect RSSI data while the robot navigating around mapped environment. The dataset, which consists of position and RSSI data pair, is then fed to our Gaussian Process model to train. The trained sensor model is capable to output sensor likelihood and the sample particles from the likelihood.

3.1 Collecting RSSI data

We would like to have our dataset to be as accurate as possible, because the dataset will be the reliable source for the learning model. The more accurate belief particle cloud mean and acceptable particle cloud variance, the more robust MCL algorithm that the learning model provides the cloud to.

In this case, **youbot** will be our robot platform to collect the RSSI dataset. Firstly,**youbot** is initialized by SSH to **youbot**'s PC.

```
roslaunch youbot_navigation youbot_init.launch
```

On the current workstation, we invoke navigation stack and open Rviz to visualize robot position on the static map.

```
roslaunch youbot_navigation static_map_amcl.launch
roslaunch youbot_navigation visualize_navigation.launch
```

Then, users instruct relatively correct position of the robot on the map (see the Figure 3) for **amcl** to reliable track robot's position online. This can be done by click on the button *2D Pose Estimate* on the top panel of Rviz then click on the map to specify robot pose.

After everything is setup, we will start probe RSSI measurements and record the data using a terminal on a laptop placed on **youbot**. These processes must be running on this laptop or else we will saturate the connection bandwidth between our computer and **youbot**.

```
roslaunch krp_localization probe_rssi.launch
roslaunch krp_localization collect_data.launch bag_name:=<your bag name>
```

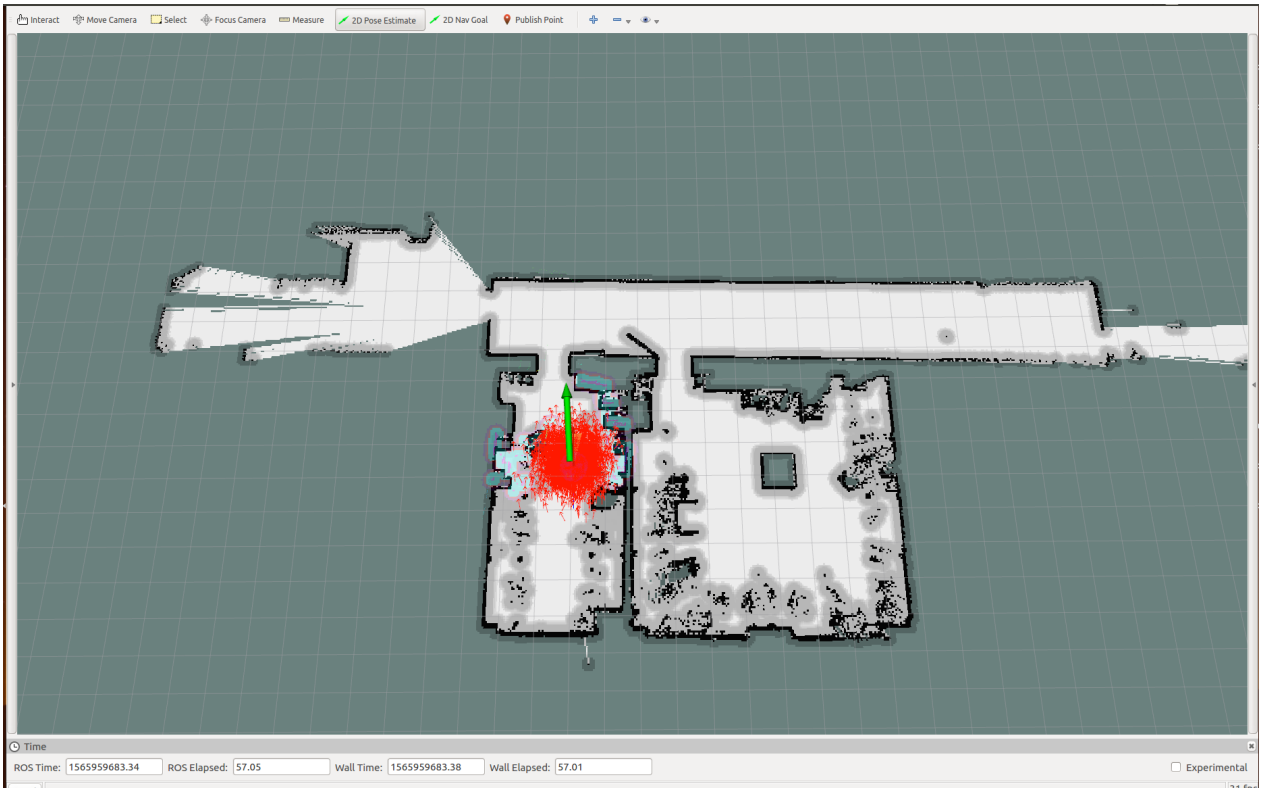


Figure 3: Initialize youbot position to start navigation

Finally, we will instruct the goals robot to navigate around the map while collecting RSSI data. The operation can be seen in the Figure 4. This can be done by click on the button *2D Nav Goal* on the top panel of Rviz then click on the map to specify goal pose. The saved data using **pickle** libraries and **rosbag** in the folder `krp_localization/data` and `krp_localization/bags`, respectively.

3.2 Affecting factors on the quality of dataset

In order for Wifi RSSI sensor model to produce reliable likelihood, the quality of training dataset is very crucial. These are factors that can affect the accuracy of training dataset:

1. Data coverage of possible position in the static map
2. Robot movement speed at the time collecting RSSI data should not be too high. (In this case, user should reduce robot translational speed)
3. Real-time factor of RSSI measurement process.
4. Number of data points.
5. Position distribution of the Wifi APs

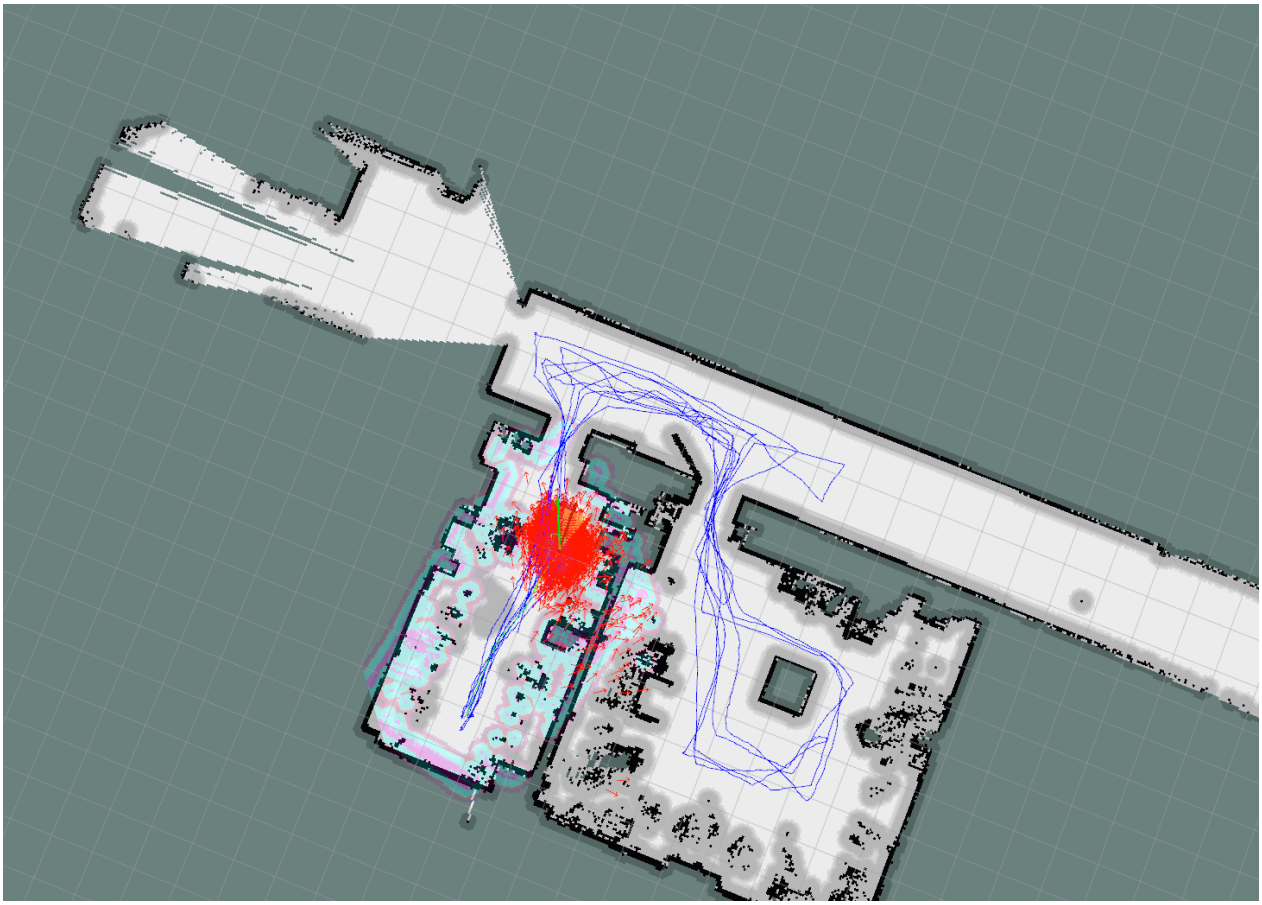


Figure 4: Recorded path as robot travelling to collect data

6. Signal coverage of each AP; some APs have very strong signal coverage, which means their signal strength do not vary in short measurement distant. This can affect prediction accuracy.

7. Number of available APs in the environment.

8. Unknown factors such as electro-disturbances, signal multipath, signal error in APs, etc.

In general, we can only control the factors from 1 to 4. With cautions and right methods, we can produce a quality dataset with high data coverage and data points. Note that user should use a separate laptop with **krp_localizaton** stack installed placed on the robot to collect RSSI measurements, in order to ensure real-time factor. Other factors from 5 to 8 we unfortunately cannot access or control. Therefore, the optimum condition to apply Wifi RSSI sensor model is to have at least 40 to hundreds APs in operation to establish data redundancy, which mitigates unknown factors' effect and have more reliable wifi signal for the sensor model.

3.3 Training models

The data collected in folder `krp_localization/data` will be use to train learning models. Users can train learning models using the `krp_localization/scripts/train_model.py` script.

User should change the variable `train_data_prefix` for the script to correctly identify new recorded data. For other parameters and flags to train the model, user should refer to Section 4.2.4. To invoke the script:

```
roscd krp_localization
python scripts/train_model.py
```

3.4 Bag Editing for simulating KRP

In real operation, it is inconvenient to manually turn off the robot sensors and carry it to other position while letting **amcl** running, in order to reproduce KRP incident. Insteads, we will record a long operation bag file, then split it into parts and merge these parts again with time gaps in between parts. With this method, we can simulate KRP incident with ease.

The functions for manipulating bag file are as followed:

- `rosbag_inspect` : shows the start and end ROS time (at the time recorded) and recorded topics.
- `rosbag_croptime` : is used to crop between user specified start and end time in bag file and output to a new bag.
- `rosbag_concatenate` : merges cropped bag file in time order with fixed transform time stamped.

These functions can be found in file `krp_localization/src/utils/data_processing.py` and an example of use is in file `krp_localization/scripts/process_bag.py` .

3.5 Model Evaluations

After training and recording bag files, we are now ready to do sanity test for the trained model. To do this, we will start playback of the bag and invoke our RSSI localization server.

```
roslaunch krp_localization playback.launch bag_name:=<enter your bag name>
roslaunch krp_localization test_rssi_localization.launch train_data_prefix:=<enter
    your data name> trained_model_path:=<enter your trained model>
```

As can be seen in Figure 5, the real robot pose is represented by blue arrow as recorded. If the green particle cloud represented RSSI model prediction, which is generated using two sampling

schemes such as Mesh Sampling and Max Gaussian Mean Sampling, can relatively track the real robot pose for the whole duration of bag file, you can evaluate that your model is trained correctly and reliable.

4 Usage

The **krp_localization** package stack has two modes of operation: playback from ROS bag files or real operation with **youbot** mobile robot.

4.1 Simulation

For simulation mode, users have to download the bags file in this embedded link and copy them to the folder `krp_localization/bags`. Users also have to export the ROS_MASTER_URI environment variable to switch roscore process back to current machine.

```
export ROS_MASTER_URI=http://localhost:11311
roslaunch krp_localization krp_localization_rssi.launch flag_simulation:=true
```

Users will see the robot invokes global localization at the beginning of the bag file and track its position through out the operation. In the middle of bag file, the robot is kidnapped to other position in the map (as can be seen on blue arrow as AMCL Reference pose), the system will detect the KRP incident and invoke KRP localization process by clearing old belief particle cloud and feeding RSSI prior particle cloud to **amcl**.

4.2 Real operation

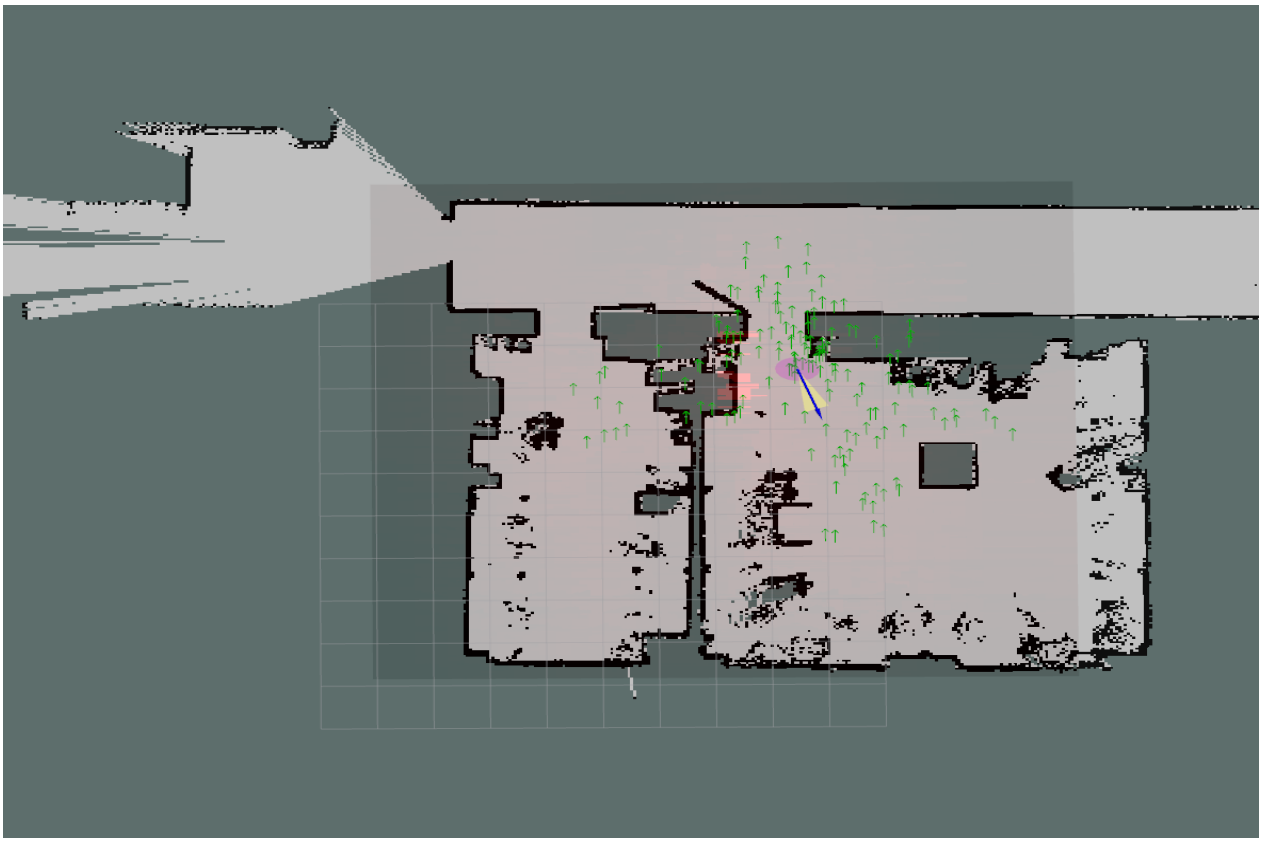
For real operation mode, users must follow these steps to setup **youbot**.

1. Plug in the battery and turn on **youbot**'s PC and motor drivers.
2. Open a terminal and SSH to youbot according to hostname specified in *Install and Config* section. Username: youbot, password: youbot

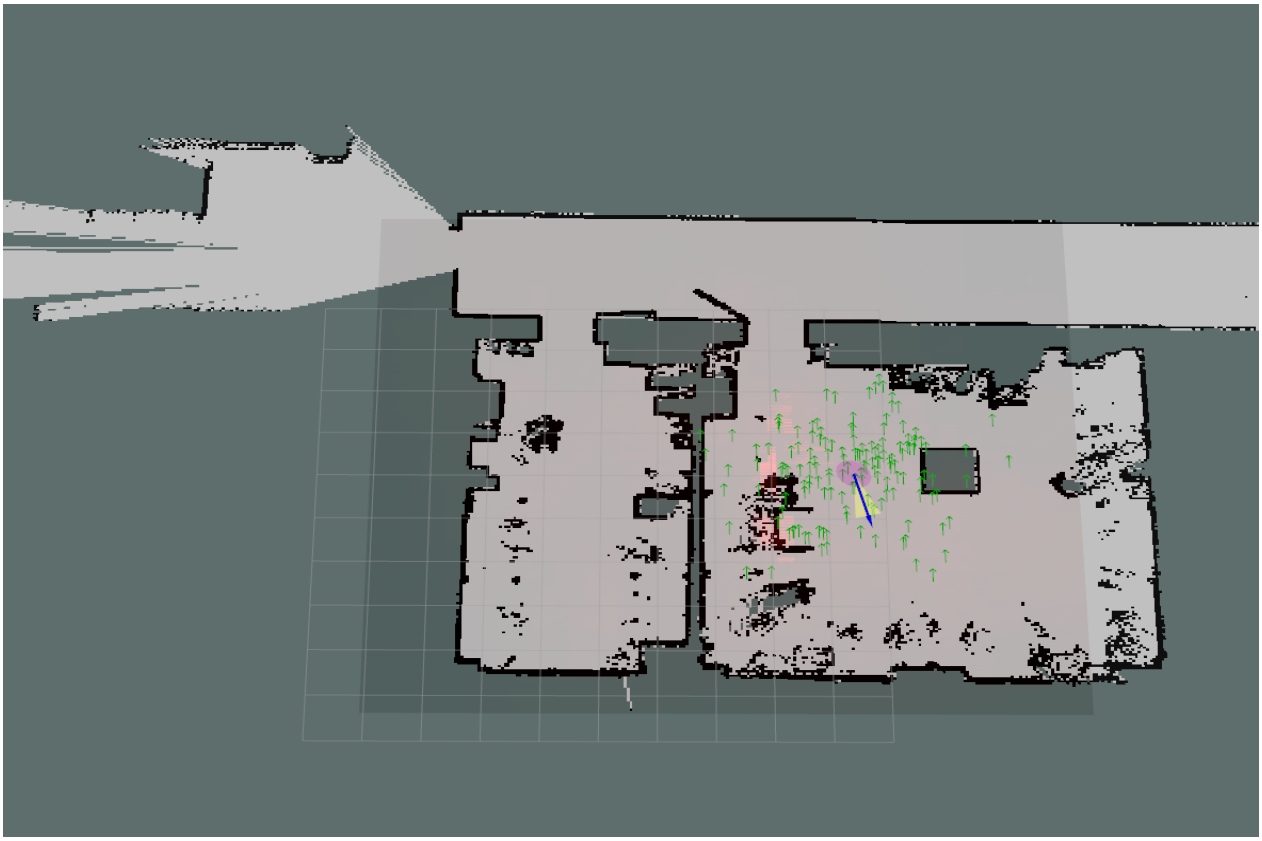
On the terminal connected to **youbot**, run the following commands to initialize **youbot**'s laser sensor, Kinect and motors.

```
roslaunch youbot_navigation youbot_init.launch
```

User will need a laptop placed on the **youbot** to probe RSSI signals due to expensive operation of WIFI signal strengths that will saturate connection between the **youbot** and workstation. This laptop is also configured followed the *Install and Config* section, then run the command:



(a) Mesh Sampling



(b) Max Gausssian Mean Sampling

Figure 5: Sanity check for RSSI belief cloud to cover current robot pose

```
roslaunch krp_localization probe_rssi.launch device_name:=<your wifi device name on
    laptop>
```

For the navigation task, open another terminal on workstation:

```
roslaunch youbot_navigation static_map_amcl.launch
```

Finally, open a terminal on workstation, type the follow command to invoke KRP mechanism:

```
roslaunch krp_localization krp_localization_rssi.launch
```

The system will then automatically localize itself based on Wifi RSSI and laser data. The navigation stack is then fully initialized and operational, user can instruct goal position on the map for the robot to proceed.

References

Le, Thai An. 2019. Approaches to solve kidnapped robot problem. Bachelor's thesis Frankfurt University of Applied Sciences.

Miyagusuku, Renato, Atsushi Yamashita and Hajime Asama. 2018. "Precise and accurate wireless signal strength mappings using Gaussian processes and path loss models." *Robotics and Autonomous Systems* 103:134 – 150.
URL: *http://www.sciencedirect.com/science/article/pii/S0921889017303925*