# CS8803: STR, Lab 1: Robot Localization

Sen Hu, Xiang Liu, Yujia Liu

February 2015

## 1 Objective

The lost robot is operating in a building with nothing but odometry and a laser rangefinder. Given a map of the building and the data from odometry and laser rangefinder, we try to help it localize itself.

## 2 Approach

To implement the particle filter (sampling with importance resampling) to localize the robot in the building, we first generate 10000 particles, $\mathcal{X}_0$, each of which represents the potential position and heading of the robot. These particles are randomly distributed in the space where the probability of unoccupied is high according to the map data. Then we perform each of the following steps.

1. Apply the odometry motion model to all particles.
2. If laser data available, calculate the posterior for each particle using the likelihood field sensor model. Otherwise, go to step 1.
3. Calculate and normalize the weights.
4. Resample particles according to their weights.
5. Go to step 1.

We first apply the odometry motion model to all particles, which will be explained more in section 2.1. Then, we have new particles with their new position and heading at the next time step. If laser range data is available in current time step, taking advantage of the sensor model, we can calculate the posterior of each particle's existence, which will leads to the weights. The sensor model is called likelihood field model and will be discussed in depth in section 2.2. After normalizing the weights, resampling takes place and induces loss of diversity. This will be explained in section 2.3.

### 2.1 Motion Model

A particle's pose at time $t$ is defined as $P_t = (x_t, y_t, \theta_t)^T$. The odometry measures the robot's motion directly on translational position and heading in unknown reference frame. However, given 2 consecutive odometry reading, its pose change can be transformed into a straight line motion(transition) $\delta_{trans}$ and a rotation $\delta_{rot}$, which are independent of the unknown reference frame. Given the odometry reading $\bar{x}_{t-1} = (\bar{x}\ \bar{y}\ \bar{\theta})$ and $\bar{x}_t = (\bar{x}'\ \bar{y}'\ \bar{\theta}')$, in an unknown reference frame, we have

$$\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2}$$
$$\delta_{rot} = \bar{\theta}' - \bar{\theta}$$

and in our global frame $\{I\}$, the motion direction is

$$\delta_{dir} = \theta_0 - \bar{\theta} + \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x})$$

where $\theta_0$ is the current heading of this particle in our global frame.

Hence, the motion between $t-1$ and $t$ in the global frame $\{I\}$ can be presented as follows

$$\begin{aligned}
\delta_x &= \delta_{trans} \times \cos(\delta_{dir}) \\
\delta_y &= \delta_{trans} \times \sin(\delta_{dir}) \\
\delta_\theta &= \delta_{rot}.
\end{aligned}$$

To model the motion error, we add each term with independent noise:

$$\begin{aligned}
x' &= x + \delta_x \times (1 + \varepsilon_1) + \varepsilon_2 \\
y' &= y + \delta_y \times (1 + \varepsilon_1) + \varepsilon_2 \\
\theta' &= \theta + \delta_{rot} \times (1 + \varepsilon_1) + \varepsilon_3
\end{aligned}$$

where $\varepsilon$ is a zero-mean guassian noise variable with variance to be tuned. The variances are intrinsic which specify the error accrued with motion. Two sources of noise are added; one that is proportional to the magnitude of the motion, and another one to simulate belief becomes less certain as time goes. These parameter variances value will be shown in section 4.1.

## 2.2 Sensor Model

For the sensor model, we use the model called likelihood field model, which can overcome the lack of smoothness and computational complexity. The end point of the laser's measurement is mapped to the global frame 180 degrees from right to left,

$$\begin{aligned}
x_{z_t^k} &= x + x_{ksen} \cos\theta + z_t^k \cos(\theta + \theta_{k,sen}) \\
y_{z_t^k} &= y + x_{k,sen} \sin\theta + z_t^k \sin(\theta + \theta_{k,sen}).
\end{aligned}$$

We assume three types of sources of noise and uncertainty: measurement noise, failures and random measurements. This results in the probability integrating all three distribution and has the following form,

$$z_{hit} \cdot p_{hit} + z_{rand} \cdot p_{rand} + z_{max} \cdot p_{max}$$

where $z_{hit}, z_{rand}$ and $z_{max}$ are intrinsic parameters.

Therefore, the resulting likelihood $q$ can be written as the product of the probability of each direction $k$ as follows

$$q = \prod_k \left( z_{hit} \cdot \text{prob}(\text{dist}_{k,t}, \sigma_{hit}) + \frac{z_{random}}{z_{max}} \right),$$

where $\text{prob}(\text{dist}_{k,t}, \sigma_{hit})$ computes the probability of $\text{dist}_{k,t}$ under a zero-centered Gaussian distribution with variance $\sigma_{hit}$. $\text{dist}_{k,t}$ is the distance to the nearest wall for one reading position and can be computed as

$$\text{dist}_{k,t} = \min_{x',y'} \{ \sqrt{(x_{z_t}^k - x')^2 + (y_{z_t}^k - y')^2} | < x', y' > \text{ occupied in m } \}.$$

### 2.3 Resampling Procedure

For particles that goes out of bound of the map, or run into a wall, they receive a likelihood of 0. Given the likelihood of all 10000 particles $q_t^i$ at time step $t$, we resample 10000 new particles randomly from the probability

$$p(i) = \frac{q_t^i}{\sum_i q_t^i}$$

where $p(i)$ is the probability that particle with index $i$ is chosen. Therefore, the particles with high weight might be reproduced multiple times and those with low weight might disappear in the next iteration, which will avoid the waste of particles in the unlikely region. The variance of the particles decreases.

## 3 Implementation

The code is written in Matlab and vectorized for fast execution. The overall algorithm has been explained in section 2. Below are a few detail implementation techniques to speed up the algorithm.

### 3.1 Efficient Posterior Calculation

To implement the sensor model in a efficient way, instead of searching in the map the closest occupied point to a given measurement from laser rangefinder in every iteration, we precompute the corresponding distance for every point in the map and store them in memory in advance. Therefore, the closest point search operation can be replaced by a (much faster) table lookup.

### 3.2 Adaptive Number of Particles

The most significant factor that determine the computational efficiency is the number of particles. With more particles to deal with in each iteration, the algorithm will be more time consuming. The trade-off between computational complexity and accuracy inspires the idea of unfixed number of particles.

The maneuver we choose is that we generate 10000 particles initially and decrease the number of particles by 2% every time there is a laser rangefinder reading coming in until 1000 particles left. Initially, the robot state is unknown, so we want plenty of particles to have all states mapped out. Since the reading from laser put more confidence on certain particles, the particles will congregate and there is no need to maintain many particles for one single position and heading. This choice works well in this localization problem as we find.

## 4 Results

### 4.1 Parameters

We assume that the noises in the odometry motion model all belong to standard normal distribution and scaled based on factor $\sigma_\varepsilon$. The parameters are shown in table 1. During testing, these parameters do not have significant effect on correctly localizing the robot. Increasing the values would cause the particles to be less concentrated.

Table 1: standard deviation $\sigma_\varepsilon$

| $\sigma_{\varepsilon_1}$ | $\sigma_{\varepsilon_2}$ | $\sigma_{\varepsilon_3}$ |
|---|---|---|
| 0.05 | 0.02 | 0.02 |

The parameters in the sensor model is shown in table 2.

Table 2: sensor model parameters

| parameter | value | tuning method |
|---|---|---|
| Initial Particles | 10000 | Enough to populate most possible initial states |
| $z_{hit}$ | 0.75 | No significant effect within reasonable range |
| $z_{rand}$ | 0.20 | No significant effect within reasonable range |
| $z_{max}$ | 0.05 | No significant effect within reasonable range |
| $\sigma_{hit}$ | 2 | Directly affect the rate the particles converges |

### 4.2 Localization Analysis

Using the algorithm, the robot is able to locate itself in the map most of the time. Video is captured on datalog 1, 3 and datalog 1 with data kidnap. The specific problem given is slightly complex because it is in a long hallway. The observation would be similar for a lot of particles until it move to certain unique position that could distinguish itself. Therefore, $\sigma_{hit}$ is set to be relative large, so the difference between particle weight would be small and they converge slowly until observation can be distinct enough.

Data kidnap is done on the first data log with 30% middle data removed. The algorithm is able to re-localize itself. The motion model compensates the missing data by adding large noise when large change in odometry data is detected. This method works well if the odometry data is consistent with the robot motion. If the robot is picked up and dropped somewhere else, then this algorithm would not work. A method is proposed in section 5 that could potentially compensate this.

## 5 Future Work & Improvement

The likelihood field sensor model deals poorly with those area that has lower probability of occupancy in the map, since we just use a threshold to distinct a object from a void. Those uncertain or unspecified areas are simply treated as unoccupied. To overcome this problem, we can extend our sensor model to sort the map value into three categories, which are occupied, unoccupied and unknown. For the unknown, we use the uniform distribution with $\frac{1}{z_{max}}$ to calculate the probability.

Another main drawback of this sensor model when implemented in the real-life situation is that it doesn't explicitly model those dynamic objects that might occur in the map, which would cause the reading to be shorter than it should be. in order to enhance the robustness of the map, we can match a map and a reading symmetrically. We can calculate the likelihood of a reading given the map and the likelihood of a map relative to a reading at the same time. This approach might efficiently rule out the dynamic objects in the map and result in a more accurate localization.

For the particle filter as a whole, one frequent problem is when the particle filter makes an incorrect prediction, it might not be able to recover using the normal techniques of prediction, weight update and resampling. Therefore, we propose to add 30% random new particles before every observation update. Sparkling the random particles through the map would helpfully capture the correct state. This may also solve data kidnapping issues.