Up Level: (parent:: Slam Framework)
#gmapping

Reference

Question

- [x] what is the map?   point position and occupied and visited value.
- [x] what map it shows in rviz? the map maintained by the particle with maximum weight.
- [x] what role does the imu play in gmapping? As an input of pose_ekf combined with the odometer to estimate the robot's positiontion
- [ ] How to calibrate the odometer? https://zhuanlan.zhihu.com/p/538697301

  /slam_course/slam14-least-squares.pptx
- [x] How to calculate the likelihood of observation?   transform laser points z to world coordinates, the point in map represent possibility.
- [x] the relation between particle weights and the close-loop? with small close-loop or small area with more features, the particle weights is concentrate more effectively.
- [x] the contour of particles.

❗ The factorization is proposal and target distribution is different.

Uploaded Files

Reference

```
/gmapping

"scanmatcher.pdf"

"RaoBlackwellized Particle Filters.3853.pdf"

"Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters.pdf"

"Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling.pdf"
```

Structure

gmapping.dot

ROS rviz flowchart

Improved distribution

Model

$$
\begin{aligned}
w_t^{(i)} &= w_{t-1}^{(i)} \frac{\eta p\left(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}\right) p\left(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1}\right)}{p\left(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}\right)} \\
&\propto w_{t-1}^{(i)} \frac{p\left(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}\right) p\left(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1}\right)}{\frac{p\left(z_t \mid m_{t-1}^{(i)}, x_t\right) p\left(x_t \mid x_{t-1}^{(i)}, u_{t-1}\right)}{p\left(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}\right)}} \\
&= w_{t-1}^{(i)} \cdot p\left(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}\right) \\
&= w_{t-1}^{(i)} \cdot \int p\left(z_t \mid x'\right) p\left(x' \mid x_{t-1}^{(i)}, u_{t-1}\right) dx'
\end{aligned}
$$

Pseudocode

if $\hat{x}_t^{(i)} =$ failure then

$x_t^{(i)} \sim p\left(x_t \mid x_{t-1}^{(i)}, u_{t-1}\right)$

$w_t^{(i)} = w_{t-1}^{(i)} \cdot p\left(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}\right)$

else

// sample around the mode

for $k = 1, \ldots, K$ do

$x_k \sim \{x_j || x_j - \hat{x}^{(i)} | < \Delta\}$

end for

// compute Gaussian proposal

$\mu_t^{(i)} = (0, 0, 0)^T$

$\eta^{(i)} = 0$

for all $x_j \in \{x_1, \ldots, x_K\}$ do

$\mu_t^{(i)} = \mu_t^{(i)} + x_j \cdot p\left(z_t \mid m_{t-1}^{(i)}, x_j\right) \cdot p\left(x_t \mid x_{t-1}^{(i)}\right.$

$\eta^{(i)} = \eta^{(i)} + p\left(z_t \mid m_{t-1}^{(i)}, x_j\right) \cdot p\left(x_t \mid x_{t-1}^{(i)}, u_t\right.$

end for

$\mu_t^{(i)} = \mu_t^{(i)} / \eta^{(i)}$

$\Sigma_t^{(i)} = \mathbf{0}$

for all $x_j \in \{x_1, \ldots, x_K\}$ do

$\Sigma_t^{(i)} = \Sigma_t^{(i)} + \left(x_j - \mu^{(i)}\right)\left(x_j - \mu^{(i)}\right)^T$

$p\left(z_t \mid m_{t-1}^{(i)}, x_j\right) \cdot p\left(x_j \mid x_{t-1}^{(i)}, u_{t-1}\right)$

end for

$\Sigma_t^{(i)} = \Sigma_t^{(i)} / \eta^{(i)}$

$// s_t$ sample new pose

$x_t^{(i)} \sim \mathcal{N}\left(\mu_t^{(i)}, \Sigma_t^{(i)}\right)$

$w_t^{(i)} = w_{t-1}^{(i)} \cdot \eta^{(i)}$

Result

Resampling

After every resampling, weight of all particles is set to be even.

```
% resample the set of particles.
% A particle has a probability proportional to its weight to get selected. A good option for such a resampling method is the so-called low variance sampling, Probabilistic Robotics pg. 109
function newParticles = resample(particles)

numParticles = length(particles);

w = [particles.weight];

% normalize the weight
w = w / sum(w);

% consider number of effective particles, to decide whether to resample or not
useNeff = false;
%useNeff = true;
if useNeff
  neff = 1. / sum(w.^2);
  neff
  if neff > 0.5*numParticles
    newParticles = particles;
    for i = 1:numParticles
      newParticles(i).weight = w(i);
    end
    return;
  end
end

newParticles = struct;

% TODO: implement the low variance re-sampling

% the cummulative sum
cs = cumsum(w);
weightSum = cs(length(cs));

% initialize the step and the current position on the roulette wheel
step = weightSum / numParticles;
position = unifrnd(0, weightSum);
idx = 1;

% walk along the wheel to select the particles
for i = 1:numParticles
  position += step;
  if (position > weightSum)
    position -= weightSum;
```

```
    idx = 1;
  end
  while (position > cs(idx))
    idx++;
  end
  newParticles(i) = particles(idx);
  newParticles(i).weight = 1/numParticles;
end

end
```

Pros & Cons

++ easy to understand, small amount of computing

— Large space is needed for each particle maintaining a map.

For example: 5 cm grid unit size to build a 200 x 200 m area, with 100 particle and 1 byte storage for grid unit data, then it need (square(200/0.05)*100*1) = 1.6 G memory size to store the map.

— Largely depend on the odometer, need accurate calibration of the sensor.

Compare with other Laser-based SLAM

-