9

Occupancy Grid Mapping

9.1> Introduction

⇒ Mapping is one of the core competencies of truly autonomous robots.

⇒ Acquiring map with mobile robots is a challenging Problem for a number of reasons:

① The hypothesis space, that is the Space of all possible maps is huge.

↳ Hence, the Bayes filtering approach that worked well for localization is inapplicable to the problem of learning maps.

② Learning map is a "Chicken-and-egg" problem, for which reason is often referred to as the Simultaneous localization and mapping (SLAM).

⇒ The hardness of the mapping problem is the result of a collection of factors, the most important of which are:

① Size ⇒ The larger the environment relative to the robot's perceptual range, the more difficult it is to acquire a map.

② Noise in perception and actuation ⇒ The larger the noise, the more difficult the problem.

③ Perceptual ambiguity → The more frequent different places look alike, the more difficult it is to establish correspondence between different locations traversed at different point in time.

④ Cycles → Cycle in the environment are particularly difficult to map.

⇒ In this chapter, we first study the mapping problem under the restrictive assumption that the robot poses are known.

9.2> The Occupancy grid Mapping Algorithm

⇒ The gold standard of any occupancy grid mapping algorithm is to calculate the posterior over the maps given the data.

$$P(m \mid z_{1:t}, x_{1:t})$$

→ The controls $u_{1:t}$ plays no role in occupancy grid mapping, since the path is already known.

→ Let m_i denote the grid cell with index i. An occupancy grid map partitions the space into finitely many grid cells.

$$m = \sum_i m_i$$

⇒ Each m_i has attached to it a binary occupancy value, which specifies whether a cell is occupied or free.

↳ Occupied = 1

↳ Free = 0

⇒ $p(m_i = 1) \iff p(m_i)$ {Probability that a grid cell is occupied}

⇒ The standard occupancy grid approach breaks down the problem of estimating the map into a collection of separate problems.

↳ Namely that of estimating $p(m_i | z_{1:t}, x_{1:t})$ for all grid cell.

⇒ This decomposition is convenient but not without problems.

↳ It does not enable us to represent dependencies among neighboring cells.

⇒ Occupancy grid mapping algorithm uses the log-odds representation of occupancy:

$$\boxed{l_{t,i} = \log \frac{p(m_i | z_{1:t}, x_{1:t})}{1 - p(m_i | z_{1:t}, x_{1:t})}}$$

⇒ Advantage: we can avoid numerical instabilities for probabilities near zero or one.

⇒ The probability are easily recovered from the log-odds ratio:

$$\boxed{p(m_i | z_{1:t}, x_{1:t}) = 1 - \frac{1}{1 + e^{l_{t,i}}}}$$

Scanned by CamScanner

Algorithm occupancy-grid-mapping $(\{l_{t-1,i}\}, x_t, z_t)$:

```
for all cells m_i do
    if m_i in perceptual field of z_t then
        l_{t,i} = l_{t-1,i} + inverse_sensor_model(m_i, x_t, z_t) - l_0
    else
        l_{t,i} = l_{t-1,i}
    endif
endfor
return {l_{t,i}}
```

⇒ The Constant l_0 is the prior of occupancy represented as a log-odds ratio:

$$l_0 = \log \frac{P(m_i)}{1 - P(m_i)}$$

⇒ The function inverse sensor model implements the inverse measurement model $P(m_i \mid z_t, x_t)$ in its long form.

α ⇒ Controls the width of the region

β ⇒ Opening angle of the beam.

~~grows~~

K ⇒ Beam Index

r ⇒ range

Algorithm inverse_range_sensor_model (i, x_t, z_t):

Let x_i, y_i be the center of mass of m_i

$r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$

$\phi = \text{atan2}(y_i - y, x_i - x) - \theta$

$k = \text{argmin}_j |\phi - \theta_{j,\text{sens}}|$

If $r > \min(z_{max}, z_t^k + \alpha/2)$ or $|\phi - \theta_{k,\text{sens}}| > \beta/2$ then

    return l_0

If $z_t^k < z_{max}$ and $|r - z_{max}| < \alpha/2$

    return l_{occ}

If $r \leq z_t^k$

    return l_{free}

endif

9.2.1) Multi-Sensor Fusion

⇒ There are two basic approaches for fusing data from multiple sensors:

① Use of Bayes filters for sensor integration.

↳ Disadvantage: If different sensor detects different type of information (obstacles), the result of Bayes filtering is ill-defined.

↳ whether or not cell is occupied depends on the relative frequency at which different sensors are polled.

③ Build separate maps for each sensor types and integrate them using the most conservative estimate.

⇒ Let $m^K = \{m_i^K\}$ denote the map build by the K^{th} sensor type.

⇒ Then the combined map is given by:

$$\boxed{m_i = \max_K m_i^K}$$

9.3) Learning Inverse Measurement Models

9.3.1) Inverting the Measurement Model

⇒ The occupancy grid mapping algorithm requires a marginalized inverse measurement model, $p(m_i | x, z)$

⇒ This probability is called "inverse" since it reasons from effects to cause.

⇒ This raises the question as to whether we can obtain an inverse model in a more principled manner, starting at the conventional measurement model.

⇒ The answer is positive but less straightforward that one might assume at first glance.

⇒ Bayes rule suggests:

$$p(m | x, z) = \frac{p(z | x, m)\, p(m | x)}{p(z | x)}$$

$$= \eta\, p(z | x, m)\, p(m)$$

Pose tells nothing about map

$$p(m_i \mid x, z) = \eta \int_{m : m(i) = m_i} p(z \mid x, m)\, p(m)\, dm$$

⇒ Clearly, this integral cannot be computed, since the space of all maps is too large.

⇒ We will now describe an algorithm for approximating this expression.

9.3.2 > <u>Sampling from the forward Model</u>

⇒ The basic idea is simple and quite universal.

⇒ If we can generate random triplets of pose $x_t^{[k]}$, measurements $z_t^{[k]}$ and map occupancy value $m_i^{[k]}$ for any grid cell m_i, we can learn a measurement function that accepts a pose x and measurement z as an input, and outputs the probability of occupancy for m_i.

(Supervised Learning Algorithm)