Robotic SLAM

1 Introduction

Terms

- State estimation - find out the pose
- Localisation - pose w.r.to landmark or map
- Mapping
- navigation and motion planning - a star, wave front dijkstra

1.1 What is SLAM

Computing robot's poses and the map of the environment at the same time.
Localisation : estimating robots location
Mapping : building a MAP

Given

- Robots control inputs

$$u_{1:T} = \{u_1, u_2, u_3....u_T\}$$

- Observations

$$z_{1:T} = \{z_1, z_2, z_3, ..., z_T\}$$

Wanted

- Map of the environment

$$m$$

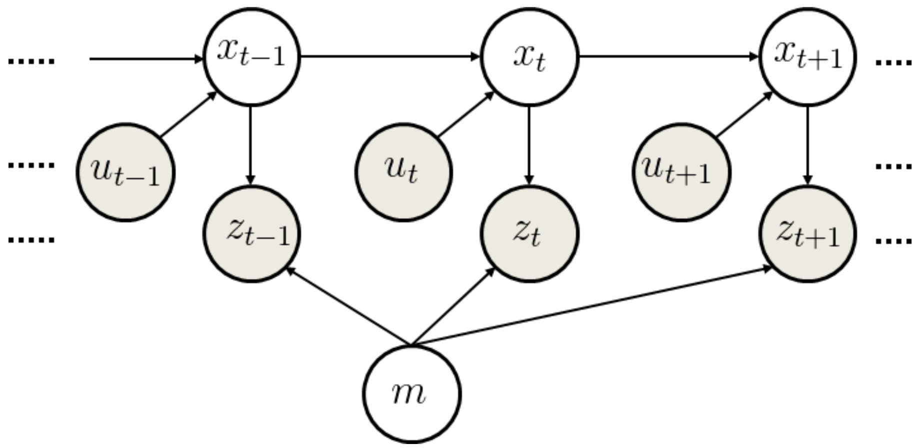
- path of the Robot

$$x_{0:T} = \{x_0, x_1, x_2, ..., x_T\}$$

Using the robots control inputs we can predict the position of the robot. From the observations $z_{1:T}$, we can calculate the position of the robot. Both the steps have some error associcated with it . Lets call the first one the model noise and second one the sensor noise. So we have to associate a probability with both of them. The error accumulates over time(even if the error in individual measurements is really small)

So in the probalistic terms our problem minimises to

$$p(x_{0:T}, m | z_{1:T}, u_{1:T})$$



1.2 Full Slam vs online SLAM

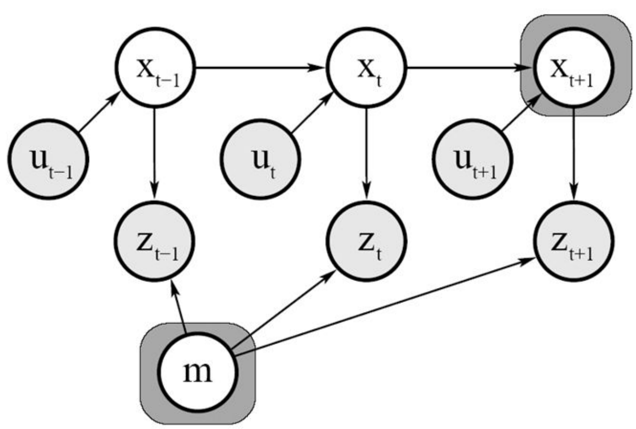
- Full SLAM estimates the entire path

$$p(x_{0:T}, m | z_{1:T}, u_{1:T})$$

- Online SLAM estimates only the most recent pose

$$p(x_t, m | z_{1:T}, u_{1:T})$$

Graphical Model of Online SLAM:



$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int\int \dots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t})\, dx_1 dx_2 \dots dx_{t-1}$$

10

1.3 Types of SLAM

occupancy maps created from lidars, sonars etc. - volumetric SLAM feature based approach - store features and localise based on that volumetric SLAM maybe better for navigation applications . Topological representations vs geometric representations. Static vs dynamic features. Active - robot decides the path so as to build a map vs passive slam - may follow a fixed path i.e. path not optimised for mapping/ exploration

2 Bayes Filter

2.1 State Estimation

Goal $p(x|z, u)$
Recursive Bayes Filter

$$\begin{aligned}
bel(x_t) &= p(x_t|z_{1:t}, u_{1:t}) \\
&= \eta p(z_t|x_t, z_{1:t-1}, u_{1:t}) * p(x_t|z_{1:t-1}, u1 : t) \\
&= \eta p(z_t|x_t) * p(x_t|z_{1:t-1}, u1 : t) \\
&= \eta p(z_t|x_t) \int_{x_{t-1}} p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t}) * p(x_{t-1}|z_{1:t-1} u_{1:t-1}) dx_{t-1} \\
&= \eta p(z_t|x_t) \int_{x_{t-1}} p(x_t|x_{t-1}, u_t) * bel(x_{t-1}) dx_{t-1}
\end{aligned}$$

we can split this into predict and update steps where
Predict Step

$$\overline{bel(x_t)} = \int_{x_{t-1}} p(x_t|x_{t-1}, u_t) * bel(x_{t-1}) dx_{t-1}$$

Update Step

$$bel(x_t) = \eta * p(z_t|x_t) * \overline{bel(x_t)}$$

Bayes filter gives a framework for recursive state estimation using the above equations. The actual realisation may be kalman filtering , EKF or particle filter (Linear non linear motion models) (distributions) *Kalman Filter* - Gaussians , requires linear or linearised model *Particle filter* - Non-parametric , Arbitrary models